UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en ingeniería informática y matemáticas

TRABAJO FIN DE GRADO

Predicción en carreras de fondo

Daniel Ruiz Mayo

Tutor: Gonzalo Martínez Muñoz

Mayo 2016

Predicción en carreras de fondo

AUTOR: Daniel Ruiz Mayo

TUTOR: Gonzalo Martínez Muñoz

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Mayo de 2016

Resumen

Este estudio se ha centrado en predecir los tiempos de finalización de carreras de atletismo de distancia maratón. Esta predicción está basada en los entrenamientos de los corredores durante sus 6 semanas previas a la prueba.

En el punto de partida de este trabajo no se disponía de ninguna base de datos ni de entrenamientos ni de carreras, por lo que para lograr el objetivo de este trabajo se ha realizado un análisis de las distintas fuentes de las que extraer la información necesaria, es decir, sus entrenamientos, su marca de maratón y las características de los corredores. La fuente por la que se ha optado después de este análisis es Strava.

Una vez elegida la fuente ha sido necesario la automatización de la descarga para conseguir el mayor número de datos posibles. La automatización es importante puesto que cuanto mayor sea el volumen de datos disponibles, mayor es el rendimiento que nos ofrecen los algoritmos de aprendizaje automático. En primer lugar la recopilación de los datos se efectuó de manera semiautomática debido a dificultades para ejecutar el Javascript de las páginas. Finalmente, tras analizar distintas posibles soluciones, la completa automatización del proceso se realizó con diversos crawlers y programas de scrapping como CasperJS.

Después de conseguir todos los datos en bruto ha sido necesaria una transformación de los datos disponibles para que los algoritmos de aprendizaje automático pudieran conseguir predicciones válidas. Este procesado de datos se ha realizado mediante programas en Java.

Posteriormente se han realizado diversos análisis y filtrados de los datos que nos han permitido mejorar la predicción de las marcas. Estos análisis han consistido en elaborar gráficas mediante scripts en R con los atributos más representativos de los corredores, con las cuales se han extraído conclusiones sobre el bajo número de entrenamientos de algunos corredores o entrenamientos anómalos. Mediante el filtrado de los corredores con información limitada se ha conseguido reducir el error de forma considerable.

Por último se ha llevado a cabo una prueba de concepto con un atleta de nuestro entorno. Se han recogido sus entrenamientos previos y su marca de maratón para, posteriormente, darles el formato que mejor nos ha funcionado durante las pruebas y que es legible por los algoritmos de aprendizaje automático. Cabe resaltar que a pesar de los problemas que el atleta tuvo en la prueba, en la que sus tiempos cayeron en los últimos kilómetros por unas pruebas médicas recientes, los resultados han sido satisfactorios.

Palabras clave

Análisis de Datos, Aprendizaje Automático, Ciencia de Datos, Clasificadores, Crawler, Entrenamiento, Maratón, Minería de datos, Regresión, Running, Scrapping, Script.

Abstract

This study focused on predicting the completion times of marathon races. This prediction is based on the runners workouts during the 6 weeks prior to the race.

In this work we did not have any database of training neither races. In order to do this an analysis on the different sources of information (meaning their training, marathon time and physical qualities) was carried out. Strava turned out to be the chosen source.

Once this was done, it became necessary to automatize the download to get as much information as possible. This automation is needed since the bigger the volume of the available data, the more accurate the algorithms output. In the beginning, data gathering was made in a semi-automatic way due to the problems caused in the page by the use of Javascript. After looking for possible solutions, automation was achieved thanks to different crawlers and scrapping software such as CasperJS.

After getting the raw data, a transformation of the information was needed so the automatic learning algorithms could achieve their predictions. This information processing was made with software in Java.

Afterwards, several analysis were carried out to help us reduce the error of the regression. These analysis consisted on figures made with scripts on R using the most representative attributes of the athletes, which allowed to draw conclusions on the low number of workouts of some runners. Through a filter of said runners we managed to considerably reduce the error.

Lastly, a proof of concept test was carried out with the help of an athlete in our environment. His previous workouts were collected, as well as his marathon record, for the purpose of formatting them in the way that resulted more efficient during the tests and that is legible by the automatic learning algorithms. It is worth highlighting the fact that in spite of the runner's problems toward the end of the race, when his records worsened due to recent medical tests, the results of the predicting algorithm were satisfactory.

Keywords

Clasifiers, Crawler, Data Analisys, Data Mining, Data Science, Marathon, Machine Learning, Regression, Running, Scrapping, Script, Training.

Agradecimientos

En primer lugar agradecer a mis padres, y al resto de la familia todo lo que han hecho por mí estos años de estudio, sin ellos nada de esto habría sido posible.

En segundo lugar agradecer a mis amigos de toda la vida, especialmente a David por sus lectura crítica, a Sara por su apoyo incondicional, su ayuda y por mi poca disponibilidad, y a mis compañeros de carrera que se han hecho imprescindibles, por aguantarme cada día.

Por último agradecer tanto a mi tutor Gonzalo, por la cantidad de horas invertidas en reuniones sobre como mejorar este proyecto, como a Ángel por ser el conejillo de indias para probar con sus entrenamientos el algoritmo de predicción.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

Una de las formas en que se percibe la creciente necesidad de sentirse en forma de toda la sociedad es el aumento en el número de aficionados a la carrera a pie, así como de inscritos en carreras amateurs o populares. El incremento de corredores responde a la necesidad de ejercitar los músculos de unos cuerpos cada vez más aletargados con motivo de las largas jornadas de oficina. No obstante, los beneficios que este deporte aporta a la salud, tanto física como mental, no son la única causa de interés que despierta esta actividad física. Se trata de un método habitual de evasión del estrés, favorece las relaciones sociales y permite tomar contacto con la naturaleza o disfrutar de un día climatológicamente propicio.

Para comprobar el auge del running solo hace falta salir a cualquier parque para comprobarlo. Pero si queremos datos objetivos, podemos comparar las 3 carreras populares que existían en Cataluña hace tres décadas con las más de 1400 que se celebran anualmente hoy en día [12]. Este aumento se observa también en el aumento del número de mujeres que practican este deporte. En 2013, casi la mitad de los 38.000 inscritos en la San Silvestre Vallecana fueron mujeres, en concreto 18.000. Otra muestra es la Carrera de la Mujer que se ha ido expandiendo por España. Actualmente se celebra en ocho ciudades y en 2014 se inscribieron cerca de 100.000 atletas.

Hay que recordar que hasta 1972 las mujeres tenían prohibido correr maratones de manera oficial. En 1967, Kathrine Virginia Switzer se apuntó a la maratón de Boston con sus iniciales, en lugar de su nombre completo, y pasó desapercibida en la inscripción, pero no en la carrera donde uno de los comisarios intentó apresarla dando lugar a una de las fotografías para la historia en cuanto a derechos de la mujer se refiere [13]. Otro dato que muestra esta desigualdad de oportunidades entre hombres y mujeres se refleja en la primera prueba olímpica de maratón. Los hombres corrieron por primera vez en las olimpiadas de Atenas de 1896. En el caso de las mujeres su primera participación se produjo en las olimpiadas de Los Ángeles de 1984.

El dato del número de corredores en España según Europapress [8] es del 10% de la población española, es decir, hablamos de unos 5 millones de personas. En el histórico de maratones de España, que se puede consultar en la Tabla 1, se puede apreciar el incremento del número de atletas que finalizan la prueba [7]. Esta Tabla nos proporciona información de los corredores que finalizan las distintas maratones de España de los años 2008 y 2013. Además nos muestra el aumento de corredores tanto en número como en porcentaje, y el número de hombres y mujeres que finalizaron las pruebas en 2013. Como podemos ver en la tabla el crecimiento del número de atletas va desde el 8% de San Sebastián hasta el 228% de Valencia. No obstante vemos como algunas maratones pierden un número mínimo de corredores. En cualquier caso, en conjunto se pasa de 28339 a 56931, es decir, un aumento en torno al 100% del número de corredores en 5 años. Esta tabla también nos proporciona una comparación con Estados Unidos, dónde el número de corredores es mucho mayor. No obstante el crecimiento en España del número participantes es considerablemente más alto. También comprobamos como el número de mujeres y hombres es mucho más equilibrado en el caso de los Estados Unidos mientras que en España en las maratones existe aún una gran disparidad.

MAR	FIN 08	FIN 13	AUM	% CRE	% APO	HOM	MUJ	% HOM	% MUJ
BARCELONA	7.634	14.776	7.142	94%	25%	12.763	2.013	86%	14%
MADRID	7.335	10.164	2.829	39%	10%	9.375	789	92%	8%
VALENCIA	2.942	9.653	6.711	228%	23%	8.770	883	91%	9%
SEVILLA	2.461	7.996	5.535	225%	19%	7.336	660	92%	8%
SAN SEBASTIÁN	2.641	2.863	222	8%	1%	2.642	221	92%	8%
MURCIA		1.668	1.668	-	6%	1.585	83	95%	5%
MÁLAGA		1.654	1.654	-	6%	1.526	128	92%	8%
CASTELLÓN		1.572	1.572	-	5%	1.482	90	94%	6%
TUI PALMA DE MALLORCA	821	1.211	390	48%	1%	935	276	77%	23%
ZARAGOZA	1.120	876	-244	-22%	-1%	836	40	95%	5%
ATLÁNTICA A CORUÑA 42		870	870	-	3%	825	45	95%	5%
COSTA DOURADA		575	575	-	2%	543	32	94%	6%
VITORIA	430	534	104	24%	0%	506	28	95%	5%
BILBAO	272	442	170	63%	1%	418	24	95%	5%
BADAJOZ	256	403	147	57%	1%	377	26	94%	6%
VIAS VERDES GIRONA		314	314	-	1%	296	18	94%	6%
EMPÚRIES	290	277	-13	-4%	0%	267	10	96%	4%
GRAN CANARIA		265	265	-	1%	250	15	94%	6%
MEDITERRÁNEO	245	240	-5	-2%	0%	216	24	90%	10%
CIUDAD REAL	240	192	-48	-20%	0%	183	9	95%	5%
LANZAROTE	208	186	-22	-11%	0%	150	36	81%	19%
CALVIÁ	99	140	41	41%	0%	130	10	93%	7%
RÍO BOEDO	42	60	18	43%	0%	56	4	93%	7%
BENIDORM	473		-473	-	-2%	-	-	-	-
TORAL DE LOS VADOS	179		-179	-	-1%	-	-	-	-
OCHOBRE	149		-149	-	-1%	-	-	-	-
SANTANDER	135		-135	-	0%	-	-	-	-
MIÑO	109		-109	-	0%	-	-	-	-
BAIXO MIÑO	100		-100	-	0%	-	-	-	-
LA RIOJA	85		-85	-	0%	-	-	-	-
HUELVA	73		-73	-	0%	-	-	-	-
ESPAÑA	**28.339**	**56.931**	**28.592**	**101%**	**100%**	**51.467**	**5.464**	**90%**	**10%**
US	**425.000**	**541.000**	**116.000**	**27%**		**308.400**	**232.600**	**57%**	**43%**
HOMOGENEIZACIÓN -									
POBLACIÓN ESPAÑA		0,12%							
POBLACIÓN US		0,17%							
EXTRAPOLACIÓN ESPAÑA		**80.461**				**45.867**	**34.594**	**57%**	**43%**
DIFERENCIA		-23.530				5.600	-29.130		
%		-29%				12%	-84%		

Tabla 1: Crecimiento del número de corredores de maratón en España

1.1 Motivación

Este trabajo nace de la mano de un profesor con gran interés por el running, la fascinación de un alumno apasionado por este deporte, y todo ello sustentado por la gran aceptación de este fenómeno deportivo por la sociedad.

Con motivo de este interés social, se plantea un proyecto de ingeniería acorde a nuestros tiempos. Nos encontramos ante un excelente marco temporal con multitud de redes sociales específicas para corredores populares con mucha funcionalidad para ver rutas, entrenamientos, resultados, etc. Además estas páginas nos ofrecen un control sobre los kilómetros, desniveles, ritmos, etc. que abarcan los entrenamientos o incluso controlar los ritmos cardíacos.

En definitiva la pasión por este deporte nos ha llevado a intentar hacer una aportación desde nuestras capacidades al mundo de la carrera a pie. En especial el objetivo principal es ayudar a corredores populares a poder tener estimaciones fiables de su posible rendimiento en maratón a partir de los entrenamientos que hayan realizado.

1.2 Objetivos

El objetivo principal de este estudio es predecir las marcas de una maratón en base a los entrenamientos de los atletas. Este objetivo se puede desglosar en los siguientes puntos:

- Analizar las webs de runners para conseguir los datos de los corredores de la web más completa.
- Analizar los datos obtenidos, procurando establecer un rango para los datos atípicos que pudieran impedir el correcto funcionamiento del proyecto.
- Clasificar a los corredores mediante algoritmos de aprendizaje automático y predecir el tiempo en una carrera de maratón en base a sus entrenamientos.
- Comparar los distintos algoritmos de aprendizaje automático para minimizar el error de regresión.
- Predecir el tiempo de un corredor externo a los datos de las webs.

Las tecnologías que se usan en este proyecto van desde el uso de crawlers para descargar la información, programas en Java para procesar los datos, uso de programas de aprendizaje automático como Weka y Knime, y programas en R para representar las gráficas de nuestros datos.

Esta memoria está estructurada del siguiente modo. En el capítulo 2 veremos el estado actual de las tecnologías actuales en lo referente a recuperación de información y aprendizaje automático. En el caso de la recuperación de información veremos las principales aplicaciones con las que conseguirlo, mientras que en el caso del aprendizaje automático trataremos conceptos generales y haremos énfasis en los principales algoritmos que usamos en este proyecto. En el capítulo 3 veremos como se ha desarrollado el trabajo y cuales han sido las fases seguidas, desde la descarga de datos, hasta el procesado de éstos. Posteriormente en el capítulo 4 veremos el análisis de los datos, así como los resultados obtenidos con los diferentes algoritmos y transformaciones de los datos. Además veremos la predicción de un corredor que se tomará como prueba de concepto. Por último en el capítulo 5 se exponen las conclusiones de este trabajo y de las posibles líneas de trabajo que se pueden seguir para continuar con el avance de este estudio.

2 Estado del arte

A lo largo de este capítulo se desarrollará la base teórica sobre la que se asienta este trabajo. Se detallarán algunos conceptos generales sobre los algoritmos de aprendizaje automático, así como conceptos generales de la minería de datos, haciendo hincapié en los conceptos y algoritmos utilizados en este proyecto.

Por otra parte se ahondará en la información recabada de los crawlers, usados en la recolección de información, parte prioritaria del proyecto para conseguir los datos de los corredores.

2.1 Recuperación Automática de Información

La Minería de Datos es el proceso de descubrir patrones y modelos descriptivos, comprensibles y predictivos a partir de datos [1]. La Minería de Datos hace uso del aprendizaje automático, a la cual está íntimamente ligada, así como a la estadística, la probabilidad y la inteligencia artificial [2].

Los algoritmos usados en Minería de Datos y aprendizaje automático son la base de la disciplina Data Science. Esta disciplina busca analizar e inferir modelos a partir de los datos para hacer predicciones. Estos algoritmos se usan desde los campos científicos hasta el campo empresarial, así como para describir comportamientos en las populares redes sociales [1].

Uno de las principales ramas de la Minería de Datos es la recuperación de información. La recuperación de información consiste en obtener elementos relevantes para el usuario a partir de repositorios masivos de información no estructurada, tal y como hacen los buscadores en la web. Los resultados de estas búsquedas son una aproximación de la información que hay en la web, puesto que hay páginas que pueden no haber sido visitadas por los crawlers. Otra rama de la Minería de Datos se encarga de evaluar cómo de buena es la aproximación devuelta por el buscador, así como de proporcionar métodos de recomendación para los usuarios. Una representación visual de la Minería de Datos la encontramos en la Figura 1.

Hay que tener en cuenta que para que un buscador pueda devolvernos los elementos relevantes a nuestra búsqueda, antes se debe rastrear la web con crawlers que recolectan información genérica de la web. Otro método para recolectar información es el scrapping que se encarga de la extracción de datos concretos de la web.

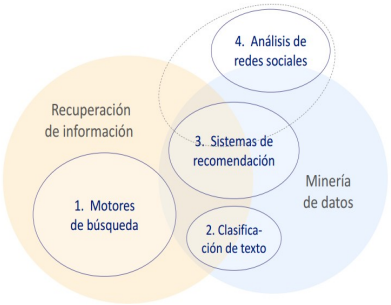

Figura 4: Minería de Datos

2.1.1 Crawlers

En esta sección nos centraremos en los Web Crawlers, ya que ha sido la manera usada para recabar la información de los corredores en la web para este proyecto.

Un Web Crawler (o araña web) es un programa que analiza páginas de la World Wide Web de forma automatizada. Uno de los usos más frecuentes de este tipo de programa es el de obtener copias de las páginas y almacenarlas de forma estructurada para el posterior procesamiento por parte de un motor de búsqueda.

El funcionamiento de estos crawlers consiste en visitar URLs en base a un patrón de forma recurrente. El crawler visita una página raíz y se guarda los enlaces y la información que aparece en ella. Estos enlaces se almacenan en una cola de prioridad o en otros tipos abstractos de datos para posteriormente poder visitarlos de forma secuencial o paralela. Para el caso de visitas de forma paralela es necesario un control sobre la lista de enlaces para no repetir las visitas y no ralentizar el proceso ni sobrecargar los servidores. Los crawlers, en general, poseen la característica de limitar la profundidad a la que se quiere llegar en la recursión de las páginas. Por ejemplo, si se pone el limite en dos sólo se permitiría acceder a los documentos que se encuentren enlazados desde la página raíz.

Las peticiones de los crawlers pueden sobrecargar los servidores tanto por el motivo anteriormente explicado como por la gran cantidad de crawlers que existen. Sin embargo, es posible para un sitio web indicar el listado de páginas a las que los crawlers no pueden acceder. Los webmaster pueden hacer uso del log con el que cuentan las webs para realizar estas peticiones. Esto es posible mediante un archivo de texto del tipo **robots exclusion standard** llamado **robots.txt**, donde se puede indicar al crawler los contenidos que puede y que no puede visitar. No obstante, los crawlers pueden saltarse este paso que es meramente consultivo e ignorar las peticiones.

En el caso de este trabajo, los crawlers se utilizan para analizar las páginas en las que se encuentran los entrenamientos de los corredores y recopilar toda la información sobre los mismos. Ahora pasamos a enumerar, describir y comparar los distintos crawlers que nos encontramos en la web. Entre los distintos crawlers privativos más usados tenemos:

- **Googlebot**: Es un bot de búsqueda desarrollado por Google. Se encarga de recolectar paginas de la web para indexar y actualizar la información de búsqueda de Google.
- **Bingbot**: Es un web-crawler desarrollado por Microsoft en 2010 para Bing similar a Googlebot. Este crawler realiza una función similar al anterior, con el objetivo de actualizar la información de Bing.

Estos Crawlers están más orientados a recolectar información genérica para estos buscadores y no tan orientados a sacar información concreta de una página, como es el caso que nos ocupa en este trabajo.

En cuanto a crawlers de software libre destaca:

- **GNU Wget:** Es un programa que forma parte de GNU, y nos permite descargar URLs vía HTTP, HTTPS, y FTP de forma sencilla. Entre sus funcionalidades más importantes está la posibilidad de descargar una página de forma recursiva y soporte para proxies. Su primera versión se lanzó en 1996. Es un programa que se usa desde la shell (línea de comandos) y está escrito en C. No obstante ha sido portado a aplicaciones gráficas como Gwget1 para GNOME, o VisualWGet3 para Microsoft.

  Una de las principales cualidades de este programa es su robustez, ya que si una descarga no se completa debido a un problema de red, Wget intentará seguir descargando desde donde acabó, y repetir el proceso hasta que la secuencia completa de enlaces haya sido recuperada. Por contra, Wget no maneja de forma correcta el Javascript de las páginas y su interpretación del código Html se limita a buscar nuevos enlaces para continuar con la recursión.

- **Import.io:** Este programa, que nace en forma de startup en Junio de 2012, nos permite descargar datos concretos de webs. A diferencia de Wget con el que descargamos el Html, con Import.io podemos descargar datos concretos que se encuentren en la página y almacenarlos en una base de datos. Sin embargo, esta aplicación sólo dispone de interfaz gráfica, es decir no podemos escribir código para adaptar mejor el programa a nuestras necesidades. Además tiene una versión web con una funcionalidad más reducida.

  Una característica positiva que nos ofrece esta aplicación es la de convertir una búsqueda de la página web, de la que queremos obtener ciertos datos, en una API sobre la que introducir otras URLs que nos vayan a devolver resultados con el mismo formato que la página anterior. La información resultante puede convertirse en un DataSet donde se encuentre toda la información extraída de estas webs similares.

2.1.2 Headless browser

Un headless browser es un navegador web que no tiene interfaz gráfica, es decir, accede a las páginas sin mostrarlo por salida externa. Estos navegadores se usan principalmente para hacer test de páginas web. Estos tests pueden ir desde determinar cómo de grandes aparecen ciertos elementos, hasta ver la fuente de letra o conocer las coordenadas x e y de un objeto en la web. Adicionalmente, se usan para extraer datos de webs o web scrapping [3]. A diferencia de los crawlers que tienen una funcionalidad más limitada, los headless browser permiten una mayor cantidad de personalizaciones para la interacción con las webs ya que son programables. Un ejemplo de headless browser es PhantomJS[11].

PhantomJS: PhantomJS es un programa de Software libre creado en el 2011 basado en WebKit, que es un entorno de navegación similar al de Google Chrome o Safari. Este programa es un navegador sin interfaz que se usa para automatizar la interacción con las webs. PhantomJS nos provee una API con la que codificar en JavaScript. Esta API permite, entre otras cosas, habilitar o deshabilitar el JavaScript de las páginas o hacer capturas de pantalla. No obstante, tiene una interactividad limitada en el uso de JavaScript.

2.1.3 Scrapping

Web scrapping es una técnica utilizada para la extracción de información de sitios web. A diferencia de los crawlers, que son utilizados principalmente para descargar páginas, las técnicas de scrapping se enfocan más en la obtención y extracción de la información que hay en la página para convertirla a algún formato más estructurado como puede ser un csv. Para realizar Scraping se utilizan una serie de programas que simulan la navegación de un usuario en la red, utilizando el protocolo HTTP o HTTPs, o incrustando en un navegador.

El web scrapping esta relacionado con la indexación de las webs aunque éste se enfoca más en transformar los datos y normalizarlos para poder insertarlos en bases de datos. También está relacionado con la automatización de procesos Web y simulaciones de visitas a webs por parte de usuarios.

Uno de los programas de web scrapping más conocido es CasperJS [10]:

CasperJS: Este programa de software libre es utilizado principalmente para realizar pruebas automáticas de sitios web mediante scripts codificados en JavaScript. Estos scripts funcionan sobre PhantomJS, el cual vimos anteriormente.

Entre las utilidades de CasperJS destaca la facilidad para hacer scrapping de webs. Esto es posible gracias a las funcionalidades que posee, tales como automatizar los clicks de una web, descargar el código HTML de una página después de cargar y ejecutar su JavaScript o hacer capturas de pantalla. Este programa complementa la funcionalidad de PhantomJS.

2.2 Aprendizaje Automático

El aprendizaje automático es una rama de la Inteligencia Artificial cuyo objetivo es inducir modelos de patrones a partir de datos sin introducir ningún conocimiento explícito en los programas. Una vez creados los modelos, éstos son utilizados para poder predecir comportamientos y clasificar datos.

El problema de la búsqueda de patrones en los datos tiene un largo recorrido en la historia. Uno de los ejemplos antiguos a destacar es el de Kepler y su descubrimiento del movimiento elíptico de los planetas, junto a sus otras 2 leyes. Estos descubrimientos se produjeron gracias a la observación de los datos y la búsqueda de patrones [4].

Veamos un ejemplo de uso de aprendizaje automático. Supongamos que tenemos una lista de datos de peces de dos especies. Supongamos que los puntos azules de la Figura 1-1 sean de la especie A y los blancos de B. El objetivo sería el de clasificar cada pez dentro de su especie teniendo datos de los atributos que los diferencien. Una posible clasificación es la de la figura 1-2.

El problema de la clasificación tiene lugar mediante varios tipos de modelado, en este caso nos vamos a basar en los dos tipos de aprendizaje más importantes, el aprendizaje supervisado y el no supervisado.

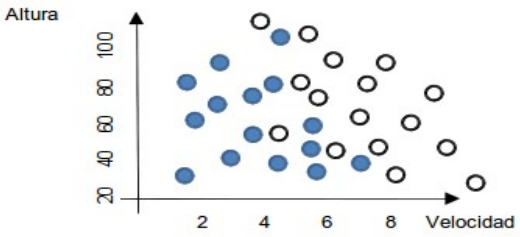

Figura 1-1:Ejemplo datos

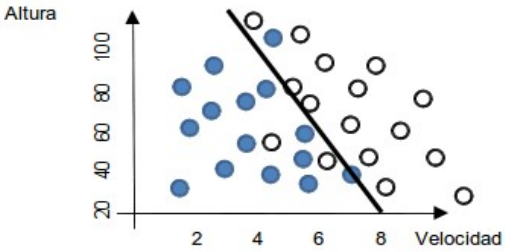

Figura 1-2:Ejemplo Clasificación

Aprendizaje no supervisado

En el aprendizaje no supervisado, el modelado de datos se realiza sobre un conjunto de datos que no tienen una clase a predecir. Los datos no están etiquetados, es por este motivo por el que se agrupan a partir de patrones inferidos en base a las características comunes de sus atributos.

Los algoritmos más usados de este tipo de aprendizaje son K-means y Density-Based Connectivity Clustering entre otros. Estos algoritmos son muy empleados y útiles en la minería social a la hora de clasificar usuarios en distintas poblaciones dentro de las redes sociales.

Aprendizaje supervisado

El aprendizaje supervisado parte de un conjunto de datos $D=\{\bar{x}_i, y_i\}_{i=1}^{N}$ donde x_i es el vector de atributos e y_i determina la clase para el ejemplo i y donde N es el número de datos disponibles. La clase es la característica principal del dato, la cual lo identifica y es lo que queremos predecir. Los atributos son las características que diferencian al ente a clasificar y nos proporcionan información con la cual predecir la clase correspondiente.

Los algoritmos de aprendizaje supervisado funcionan haciendo suposiciones sobre los datos para poder generar un modelo que los explique. Hay que diferenciar entre modelos de clasificación y de regresión: la clasificación está orientada a predecir el valor de una clase discreta, mientras que esta última está enfocada a predecir el valor de una variable continua.

Los datos sobre los que se construye el modelo están formados por ejemplos clasificados. Un ejemplo de esto es la clasificación de las dos variedades de peces explicada anteriormente. En este caso la especie es la clase y el peso, medidas y color, los atributos que los distinguen.

Para comprobar la efectividad de los algoritmos de aprendizaje automático es importante definir medidas significativas. Para el caso de regresión, que es el que usamos en este trabajo, las métricas que consideramos más importantes para medir la precisión de los modelos son el error absoluto y el error cuadrático medio. El error absoluto se mide como $E_{\text{abs}}=\frac{1}{n}\sum_{1}^{n}|\hat{Y}-Y|$ siendo $\hat{Y}$ el valor de la predicción e Y el valor real. El error cuadrático medio da un mayor peso a los errores con grandes diferencias entre el estimado y el real. Este error se define de la siguiente manera $E_{CM}=\frac{1}{n}\sum_{1}^{n}(\hat{Y}-Y)^2$.

En este proyecto son muy importantes los modelos de clasificación supervisados, es por esto que pasamos a describirlos a continuación. No obstante, debido a la gran cantidad de algoritmos, vamos a describir sólo los considerados más representativos.

2.2.1 Naive Bayes

El algoritmo de clasificación de Naive Bayes se basa en la regla de Bayes:

$$P(H|x)=\frac{P(x|H)\cdot P(H)}{P(x)}$$

que nos permite obtener la hipótesis más probable H dado un vector de atributos x, es decir, hay que calcular:

$$P(H_i|\bar{x})\,\forall\, i=1,\ldots,k$$

Aplicando el teorema de Bayes y expandiendo el vector de atributos tenemos:

$$P(H_i|\bar{x})=\frac{P(\bar{x}|H_i)\cdot P(H_i)}{P(\bar{x})}=\frac{P(x_1,\ldots,x_n|H_i)\cdot P(H_i)}{P(x_1,\ldots,x_n)}$$

La complejidad del cálculo de esta ecuación crece exponencialmente con el número de atributos. Para evitar esto, el método Naive Bayes [4] supone que los atributos son independientes entre sí dada la clase, es decir:

$$\frac{P(x_1,\ldots,x_n|H_i)\cdot P(H_i)}{P(x_1,\ldots,x_n)}=\frac{P(x_1|H_i)\cdot\ldots\cdot P(x_n|H_i)\cdot P(H_i)}{P(x_1,\ldots,x_n)}$$

lo que reduce la complejidad del cálculo a un crecimiento lineal con el número de atributos. Para calcular $P(x_1,\ldots,x_n)$ podemos utilizar:

$$P(x_1,\ldots,x_n)=\sum_i \prod_j P(x_j|H_i)\cdot P(H_i)$$

Ahora ya podemos construir la regla de clasificación con Naive Bayes del siguiente modo:

$$H^*=argmax_{H_i}\prod_i^n P(x_j|H_i)\cdot P(H_i)$$

La complejidad en tiempo de clasificación de este método es O(k·n), donde n es el número de atributos y k el número de clases.

2.2.2 Regresión lineal

La regresión lineal es un modelo matemático que se usa para cuantificar la relación de dependencia lineal entre una variable de regresión *Y,* dependiente, y las variables independientes regresoras x_i . Estas variables tienen asociadas un término de error aleatorio ε.

Este modelo se puede expresar de la siguiente manera:

$$Y_i=B_0+B_1\cdot X_1+B_2\cdot X_2+\ldots+B_n\cdot X_n+\varepsilon_i, i\in 1\ldots n$$

siendo los B_i los parámetros que cuantifican la influencia de cada variable regresora en la variable de regresión y donde la variable de error ε_i verifica que

- ε_i tiene media cero $\forall i$.
- Var(ε_i) $=\sigma^2$, $\forall i$, condición de homocedasticidad.

Para estimar los parámetros del modelo, podemos usar el criterio de mínimos cuadrados:

$$\|Y-X\cdot B\|^2=\sum_{i=1}^n [Y_i-(B_0+B_1\cdot X_{i1}+B_2\cdot X_{i2}+\ldots+B_k\cdot X_{ik})]^2$$

donde se verifica que los valores de $\hat{B}_0,\hat{B}_1,\ldots,\hat{B}_k$ que minimizan la ecuación son:

$$\hat{B}=(X^t\cdot X)^{-1}\cdot X^t\cdot Y$$

Además, todos los estimadores $\hat{B}_j$ verifican que $\frac{\hat{B}_j - B_j}{error\,típico\,\hat{B}_j}$ sigue una distribución t de Student con n-k-1 grados de libertad.

2.2.3 Vecinos Próximos

Vecinos próximos es un algoritmo que consiste en clasificar un nuevo ejemplo basándose en la clase de los ejemplos cercanos. En concreto en este algoritmo se fija el número de vecinos, K, y se calcula una esfera alrededor del punto de interés hasta que incluya K puntos [4]. La fase de entrenamiento de este algoritmo consiste en guardar todo el conjunto de entrenamiento D.

En concreto para clasificar un ejemplo z, el algoritmo funciona con los siguientes 3 pasos:

1. Por cada ejemplo de los datos, x_i, se calcula la distancia de z a x_i usando la distancia elegida.
2. Se seleccionan los K datos más cercanos.
3. Se devuelve la clase más común en esos K ejemplos si se trata del caso discreto, o la media de los K si es una regresión. Otras versiones alternativas consisten en ponderar la clase de los K puntos más cercanos en base a la distancia, de forma que los puntos que están más cercanos influyan más en la predicción.

En este algoritmo es crítico el cálculo de distancias. Las métricas más utilizadas para calcular las distancias entre puntos son la euclídea:

$$d(\bar{x}, \bar{y}) = \sqrt{\sum_i^n (x_i - y_i)^2}$$

y la de Mahalanobis, que tiene en cuenta la variabilidad de los datos:

$$d(\bar{x}, \bar{y}) = \sqrt{\sum_i^n ((\bar{x} - \bar{y}))^t \cdot \lambda^{-1} (\bar{x} - \bar{y})}$$ Siendo λ la covarianza.

La complejidad en tiempo de clasificación de este algoritmo es O(N · #{ atributos })

2.2.4 Bagging

Bagging es un algoritmo de clasificación del tipo de conjuntos de clasificadores que se basan en combinar los resultados de varios modelos [9]. Bagging es el acrónimo de bootstrap agregation, y es uno de los más populares dentro de los algoritmos de conjuntos de clasificadores. Este algoritmo reduce la varianza en la salida de los clasificadores que combina. Además es un método que no incurre en sobreajuste. Esto quiere decir que no sobreentrena el algoritmo con los datos de los que se conoce el resultado. Esta característica es importante para poder generalizar y resolver situaciones distintas a las conocidas de la fase de entrenamiento. La ventaja principal de

bagging, y de los conjuntos de clasificadores en general, es que al promediar varios modelos se obtiene un resultado mejor que cuando se usa un sólo modelo.

Este algoritmo genera un conjunto de modelos repitiendo el siguiente proceso: Generar una muestra bootstrap de los datos y construir un modelo. Una muestra bootatrap, Di, consiste en extraer N datos con reemplazamiento del conjunto D que tiene tamaño N. La muestra Di contendrá datos repetidos de los datos originales D y algunos que no tendrá. En media tiene un 63% de los datos originales. Esta variabilidad en los datos de entrenamiento hace que los modelos sean diversos [6]. A la hora de clasificar se hace la moda de las decisiones de los modelos en clasificación y la media para regresión.

El algoritmo conlleva los siguientes pasos para entenarlo:.

1. For i=1 hasta T modelos
2. D_i MuestraBootstrap(D).
3. Los datos D_i se usan para entrenar un modelo Mi.
4. Return todos los Mi

2.2.5 Aplicaciones

En cuanto a las herramientas representativas de aprendizaje automático encontramos Weka y Knime.

Weka

Esta aplicación nos provee mediante una sencilla interfaz de un gran conjunto de algoritmos de aprendizaje automático listos para ser usados con archivos que almacenan los conjuntos de datos. Este programa nos permite configurar los parámetros de los algoritmos que vamos a utilizar. Los algoritmos están programados en Java y están en código abierto, por este motivo de manera sencilla podemos modificar los algoritmos a nuestra elección. Los algoritmos que integra esta aplicación son tanto de aprendizaje supervisado como no supervisado.

A la hora de predecir, el programa nos informa de los errores medios y cuadráticos, así como del coeficiente de correlación. Además, Weka nos proporciona visualizaciones de los histogramas y gráficas de correlación de las distintas variables. Los archivos con los que trabaja Weka disponen de un campo que informa del nombre y del tipo de dato de cada atributo.

Knime

Este programa de aprendizaje automático nos ofrece la posibilidad de usar los algoritmos de Weka, así como otros algoritmos propios de la aplicación. Asimismo es una ETL, Extract Transform and Load, ya que ofrece métodos para extraer los datos tanto de fichero como de bases de datos, así como transformarlos o mostrar la información oportuna en gráficas y extraer predicciones a fichero. Es un programa muy completo y visual como podemos ver en la Figura 2, esto nos permite seguir el flujo y moldearlo a nuestras necesidades. En este ejemplo se pueden ver funcionalidades básicas como extraer datos de un fichero Arff, entrenar modelos, realizar predicciones o escribir a fichero csv.

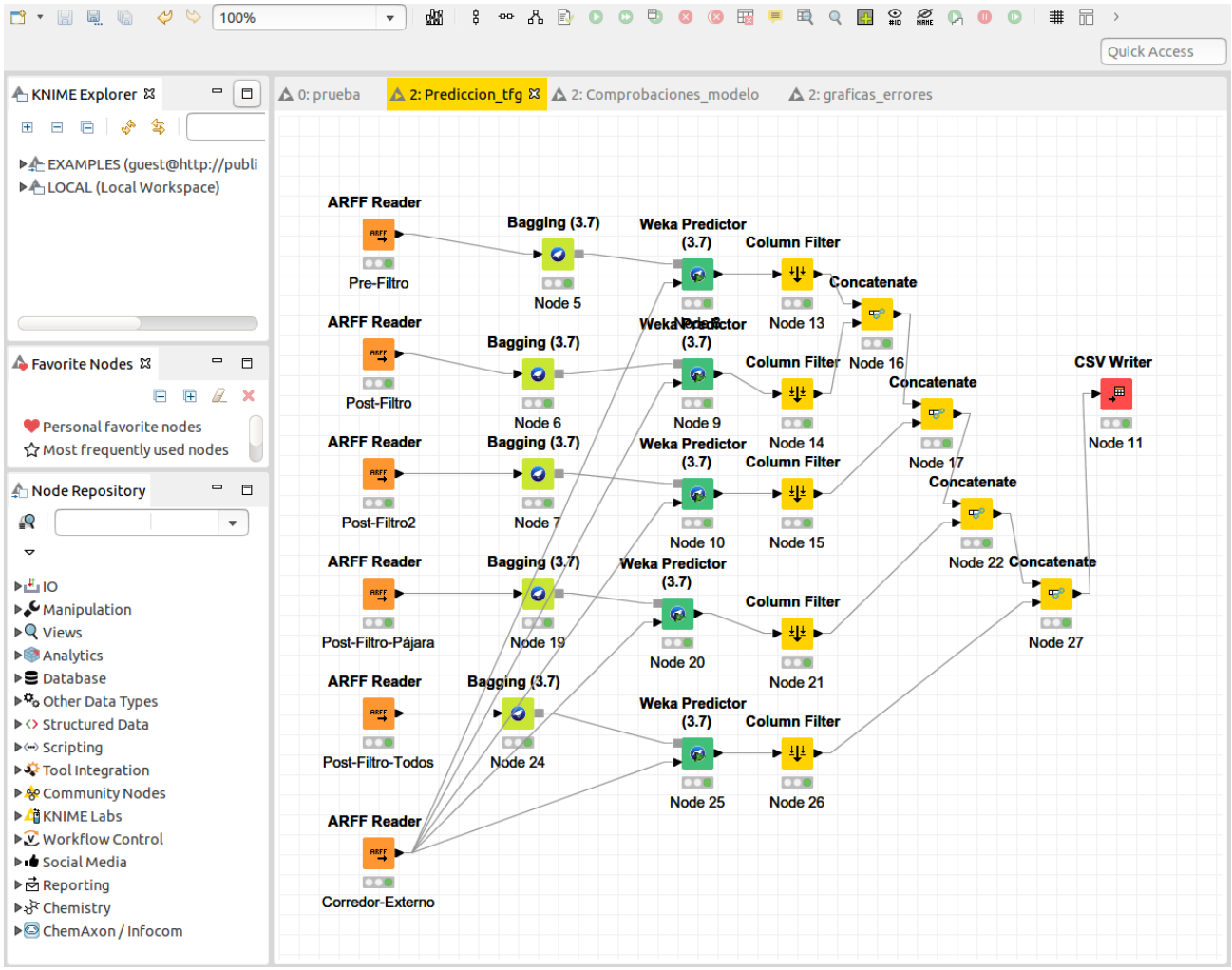

Figura 2: Ejemplo gráfico Knime

Una de las cualidades positivas además de las ya mencionadas es la posibilidad de comparar los resultados, así como de observar de forma sencilla los fallos en la predicción mediante un simple botón. Esto nos aporta sencillez y funcionalidad para evaluar la efectividad de los clasificadores.

3 Desarrollo

El objetivo de este trabajo es predecir el tiempo que va a conseguir un corredor en base a sus entrenamientos previos. Para lograr nuestro objetivo es importante conseguir la mayor cantidad de entrenamientos clasificados posibles. Es importante disponer de la marca en el maratón ya que los algoritmos entrenan con los corredores conocidos y en base a sus entrenamientos y marca predicen la marca de los nuevos atletas. Además nos encontramos en un marco temporal inmejorable para la realización de este estudio. Esto es debido a la gran cantidad de deportistas que encontramos hoy en día y a la multitud de páginas existentes en las que los corredores pueden subir sus entrenamientos.

Para poder desarrollar este trabajo se ha recopilado información sobre los entrenamientos de los corredores. Antes de extraer estos datos se realiza un análisis previo de la situación en la web de las distintas páginas y aplicaciones en la que los atletas comparten tanto sus entrenamientos como sus carreras.

Por un lado se compararon las plataformas Runkeeper y Nike+ running. Ambos están disponibles en versión móvil, por lo que era muy complicado descargar sus datos. Adicionalmente tanto Runkeeper como Nike requieren de una amistad previa de los usuarios para mostrar los datos de sus carreras y entrenamientos. Por estos motivos se descartan ambos ya que para automatizar este proceso es necesaria una mayor flexibilidad.

Por otro lado se analizaron Endomondo y Strava. Estas plataformas tienen tanto aplicación para smartphone como versión web. Endomondo al igual que Nike y Runkeeper nos solicitan una amistad previa para revisar los entrenamientos y además Endomondo no dispone de una página exclusiva en la que consultar los participantes de las carreras de fondo. Por el contrario, en Strava disponemos de una página en la que acceder a las pruebas donde ver las marcas de los corredores. Además esta página nos proporciona un enlace a sus perfiles, facilitando la automatización de la extracción de datos. Otro punto a favor de Strava es que no es necesario una relación de amistad entre los usuarios para acceder a los perfiles. Adicionalmente tanto en Strava como Endomondo se permite un filtrado temporal. Esto es muy útil para este proyecto, puesto que solo vamos a analizar los entrenamientos de una fecha previa determinada a la maratón.

Por todos los motivos anteriores, se decide optar por la única página que disponía de la información de carreras y entrenamientos de manera pública y que proporcionaba un acceso más sencillo a la información: Strava.

Se ha considerado recoger datos y estadísticas de las 6 semanas anteriores a la maratón por considerar que son las más representativas. Hay que tener en cuenta que para entrenar una maratón son necesarias unas 12-16 semanas según los programas convencionales [14]. Por este motivo se consideran suficientes las últimas 6 semanas que nos aportan una visión más precisa del estado de forma de los corredores en el periodo previo a la competición. Un detalle a tener en cuenta es que las primeras semanas se consideran una prueba de contacto con los entrenamientos mientras que las últimas nos dan una información más precisa y fiable para la prueba.

Ahora vamos a detallar las fases del proceso que seguimos en el tratamiento y análisis de los datos brutos hasta encontrar una predicción satisfactoria de los tiempos de las carreras.

3.1 Fase 1: Extracción semiautomática

En una primera fase, la extracción de los datos se llevó a cabo de manera semiautomática para las carreras de Sevilla y de Barcelona. La extracción de los datos de los corredores y sus entrenamientos se hizo en 2 etapas: Obtención de los datos básicos de los corredores y sus marcas en maratón y descarga de sus entrenamientos. Esta segunda etapa se realiza de forma semiautomática.

Para la primera etapa se utiliza la aplicación Import.io. Mediante una sencilla interfaz se consiguen los atributos necesarios de los corredores:

- Ritmo (minutos por kilómetro de media)
- Tiempo de finalización de la maratón
- Edad
- Sexo

Además de estos atributos se consigue una URL a los entrenamientos del corredor. Esta página se caracteriza por un un panel con estadísticas del atleta, un filtro temporal para las fechas de los entrenamientos y una lista con los entrenamientos de las fechas seleccionadas. Aquí empieza el proceso manual, que consiste en conseguir el HTML de esta URL con los entrenamientos del corredor correspondientes a las 6 semanas previas a la prueba. En un principio no se automatizó debido a que la página requería JavaScript para recargar los entrenamientos de las 6 semanas anteriores a la maratón y, por este motivo **`wget`** no era capaz de aportar la información necesaria. Este programa únicamente permitía descargar los entrenamientos de la semana actual pues los anteriores requerían de una interacción de JavaScript que se desencadena mediante un click.

Por otra parte, para que este script funcionara correctamente había que estar previamente logueado en la página web de Strava. El método usado para descargar estos atributos era usar el crawler WGet de Linux. Para poder realizar el login con **`wget`** se usa un fichero de cookies, exportado mediante la herramienta de Firefox Export Cookies. Mediante un parámetro se indica al script la ruta de las cookies. De esta manera damos a entender a la página que estamos logueados con el inicio de sesión que hemos completado previamente en nuestro navegador antes de exportar la cookie.

No obstante, **`wget`** tiene unas cualidades que facilitaron la descarga de información, como se ve en **2.1.1 Crawlers**. Estos beneficios consisten en la robustez que ofrece **`wget`**. Al encontrarse con descargas fallidas por motivos de red, vuelve a intentar la descarga hasta que finaliza con éxito. Esta ventaja es importante para este estudio debido a que no es deseable extraviar entrenamientos de los corredores pues esto podía provocar inconsistencias en el proyecto empeorando los resultados.

Después de conseguir el HTML con los entrenamientos, un script de bash, que se puede consultar en **A.1 Script Bash**, se encargaba de analizar el HTML para extraer las URLs de cada entrenamiento y sacar la información relevante de éstos. Este proceso tiene dos puntos críticos.

Por una parte, parsear los HTML, esto es, transformar toda la información que nos llega separando solamente los datos relevantes. Esto se realiza con comandos básicos de UNIX como **`grep`**, **`cut`**, **`sed`** y **`head`** entre otros. Una de las funcionalidades básicas necesarias fue omitir los entrenamientos en bicicleta, realizado con un filtro a base de comandos **`grep.`**

Todo esto ha permitido recuperar la información necesaria. Esta información se puede resumir en los siguientes atributos:

- Tiempo del entrenamiento
- Ritmo (minutos por kilómetro de media)
- Distancia en kilómetros del entrenamiento
- Desnivel acumulado en metros

Tras estos procedimientos se procede a la limpieza y formateo de los datos (más detalles en **3.4 Limpieza y Formateo de los Datos**). Cabe resaltar que por tratarse de un método semiautomático, el número de corredores de estas carreras es bajo.

3.2 Fase 2: Automatización

Cuanto mayor es el número de datos de los que disponemos, mayor precisión obtenemos con los algoritmos de aprendizaje automático y mejor consiguen aproximarse a la realidad. Por este motivo se opta por investigar de qué manera automatizar completamente la descarga de los HTML de los entrenamientos de Strava. Para ello es necesario que se desencadene la ejecución del Javascript de forma automática. La solución que hemos adoptado utiliza CasperJS basado en PhantomJS. Este programa nos permite interactuar con el JavaScript de la página. La información detallada de estos programas se encuentra en **2.1.2 Headless browser** y en **2.1.3 Scrapping**.

Se crea un script de CasperJS con el que conseguimos tratar el Javascript necesario para visualizar los entrenamientos. Con un logueo previo en la página con comandos de CasperJS conseguimos descargar el HTML buscado. El login es necesario, puesto que para acceder a la información es imprescindible haber iniciado sesión.

Previo a la finalización con éxito de este script se hace uso de la funcionalidad de capturas de pantalla que ofrece CasperJS, que nos permite controlar si la página ha cargado correctamente. Este script se puede consultar en **A.2 Script CasperJS**. Este script se integra en la primera fase sustituyendo la parte semiautomática. De esta manera se automatizó la descarga de los entrenamientos y marcas de las maratones de Boston, Los Ángeles y Londres

3.3 Fase 3: Filtrado

Tras descargar todos los entrenamientos se comprobó que había muchos atletas con muy pocos entrenamientos. Esto sorprende ya que para preparar de forma adecuada una maratón, se recomiendan al menos 3 entrenamientos a la semana, además de ser constantes en el esfuerzo. Estos resultados se deducen de las gráficas que veremos más adelante en la sección **3.6 Extracción de atributos**. Por este motivo, se procede al filtrado de la primera etapa de los usuarios que no cumplen con los requisitos. Estos requisitos consisten en realizar al menos 3 entrenamientos en cada semana. Para este requisito no se ha tenido en cuenta la última semana,

ya que se considera de reposo y puede haber corredores que hayan realizado un número menor o incluso no hayan entrenado.

Tras el anterior filtrado se comprueba la existencia de algunos datos poco coherentes. Estos datos son de entrenamientos de más de 50 kilómetros. Se filtran estos entrenamientos en la segunda etapa para evitar posibles errores en la clasificación. Dado el gran número de kilómetros puede tratarse de un entrenamiento en bicicleta mal identificado por el usuario. Adicionalmente, se filtraron los entrenamientos con un ritmo menor de 2 minutos por kilómetro, por razones que se detallan en profundidad en **4.2 Resultados Post-Filtro**.

3.4 Limpieza y Formateo de los Datos

Estos datos en crudo correspondientes a los entrenamientos de los atletas junto con sus marcas, tienen que ser procesados mediante dos programas en Java. Esto es necesario puesto que los algoritmos de aprendizaje necesitan que todas las observaciones tengan el mismo número de atributos.

El primer procesado se encarga de normalizar los entrenamientos. Los entrenamientos que se encuentran sin el atributo desnivel se rellenan con 0, mientras que si falta el tiempo o la distancia se borran, puesto que no son registros válidos.

Los entrenamientos en los que no hay desnivel tienen una explicación sencilla: hay corredores que llevan un pulsómetro, suben esa información a la web y por motivos internos de Strava, la página deja de mostrar el desnivel para que aparezca la frecuencia cardiaca.

El segundo procesado se encarga de transformar estos datos a formato Arff, de modo que sean legibles por Weka. Este procesado se realiza mediante programas codificados en Java. Se almacena una estructura "corredor", que tiene entrenamientos de los periodos determinados en **3.6 Extracción de atributos**. El programa hace el correspondiente almacenaje y transformación.

3.5 Datos

Los datos descargados incluyen entrenamientos previos a las siguientes maratones:

- Maratón de Barcelona fecha: 15 de marzo de 2015
- Maratón de Boston fecha: 20 de abril de 2015
- Maratón de Sevilla fecha: 22 de febrero de 2015
- Maratón de Los Ángeles fecha: 15 de marzo de 2015
- Maratón de Londres fecha: 26 de abril de 2015

Los datos de las diferentes maratones se pueden observar en las Tablas 2 y 3. Ambas tablas tienen la misma estructura, la maratón de la que se trata, número de corredores, número de entrenamientos, número de entrenamientos por corredor, número de hombres, número de

mujeres y número de corredores del que no disponemos de información de género por no haberla proporcionado estos usuarios a la aplicación. Estos corredores sin género aparecen marcados con una '?'.

En la Tabla 2 vemos los datos de los corredores antes de las cribas. Podemos observar la disparidad de entrenamientos por corredor entre Barcelona y Los Ángeles respecto del resto de carreras, es por esto que se procede al filtrado. El bajo número de corredores de Barcelona y Sevilla se debe a que el proceso no estaba automatizado.

Carreras	Corredores	Número de entrenamientos	entrenos/corredor	Hombres	Mujeres	?
Barcelona	47	935	19,89	41	6	0
Boston	1620	52304	32,28	1235	81	304
Los angeles	625	12117	19,38	451	174	0
Sevilla	70	1978	28,25	70	0	0
Londres	1766	52626	29,79	1484	282	0

Tabla 2: Estadísticas corredores Pre-Filtro

En la Tabla 3 podemos ver los datos de los corredores una vez aplicado el filtrado de la **3.3 Fase 3: Filtrado**. Si comparamos con la Tabla 2, vemos que hay un aumento considerable del número de entrenamientos por corredor. Además conseguimos que se homogeneicen.

Carreras	Corredores	Número de entrenamientos	entrenos/corredor	Hombres	Mujeres	?
Barcelona	15	444	29,6	15	0	0
Boston	792	27068	34,17	645	45	102
Los angeles	89	3642	40,92	75	14	0
Sevilla	21	719	34,23	21	0	0
Londres	208	8541	41,06	190	0	18

Tabla 3: Estadísticas corredores Post-Filtro

3.6 Extracción de atributos

Como se ha explicado en **3.4 Limpieza y Formateo de los Datos**, para que los algoritmos de aprendizaje automático funcionen se necesita un mismo número de atributos en todas las observaciones. Por este motivo no podemos usar los datos en su formato original. De esta manera creamos atributos para que todos los corredores sean homogéneos. Los atributos serán estadísticas representativas de los entrenamientos en periodos de tiempo. Se han definido distintos periodos de tiempo que dan lugar a 3 conjuntos de atributos distintos para los entrenamientos de las 6 semanas anteriores a la maratón. Los periodos considerados tratan los entrenamientos por semana, cada 2 semanas, y de forma aglomerativa.

El aglomerativo consiste en tener los entrenamientos organizados en 6 periodos semanales, 3 periodos de dos semanas, y un periodo de todos los entrenamientos. Donde el semanal consiste en 6 periodos semanales mientras que el de 2 semanas son 3 periodos de 2 semanas cada uno.

Cada periodo temporal se compone de:

- Kilómetros del entrenamiento con mayor distancia del periodo
- Kilómetros del entrenamiento con el ritmo más veloz del periodo
- Kilómetros totales de los entrenamientos del periodo
- Tiempo del entrenamiento con mayor distancia del periodo
- Tiempo del entrenamiento más rápido del periodo
- Tiempo total de los entrenamientos del periodo
- Desnivel acumulado del entrenamiento con mayor distancia del periodo
- Desnivel acumulado del entrenamiento más rápido del periodo
- Desnivel total acumulado de los entrenamientos del periodo
- Ritmo del entrenamiento con mayor distancia del periodo
- Ritmo del entrenamiento más rápido del periodo

En resumen para el conjunto de datos semanal habrá 11x6 atributos, para el bisemanal 11x3 y para el aglomerativo 11x9.

3.7 Visualización de datos

Una buena manera de analizar los datos es mediante visualización. En la Figura 3 se muestra una tabla con 6 histogramas que hacen referencia a las 6 maratones mencionadas en **3.5 Datos** y al conjunto de todas ellas. Cada histograma muestra el número de corredores por cada franja de tiempos de la maratón.

Como podemos comprobar en los tiempos de la gráfica correspondiente a todas las maratones juntas, los tiempos se concentran en mayor medida entre los 10800 segundos y los 15000, esto traducido en horas va de las 3 horas a las 4 horas y 10 minutos. Los histogramas de Barcelona y Sevilla aparecen juntos debido al bajo número de corredores por tratarse de la primera versión no automatizada. Cabe destacar que para la maratón de Boston se observa un desplazamiento de la distribución hacia la izquierda con tiempos inferiores al resto de carreras. Esto es así porque para correr esta maratón se requiere de la acreditación de un tiempo mínimo. Los hombres de entre 18 y 34 años de edad deben acreditar un tiempo inferior o igual a 3h05, en un maratón oficial del año anterior al de inscripción. Si nos fijamos en el pico de tiempos se sitúa entorno a los 11500 segundos, o lo que es lo mismo, 3 horas 10 minutos.

Otro aspecto relevante de estos datos es que parece que son resultados más competitivos que la media de los que se observan en maratones populares. Para comprobar esto vemos por ejemplo la mediana de tiempos de la maratón de Madrid de 2013, que está en **3:51:24**. Como podemos comprobar en todos los datos de las maratones descargadas este valor es inferior. La mediana de tiempos de todas las maratones es de **3:33:12**. Este pequeño sesgo se podría explicar por una posible mayor motivación de los corredores que suben sus entrenamientos a las plataformas digitales.

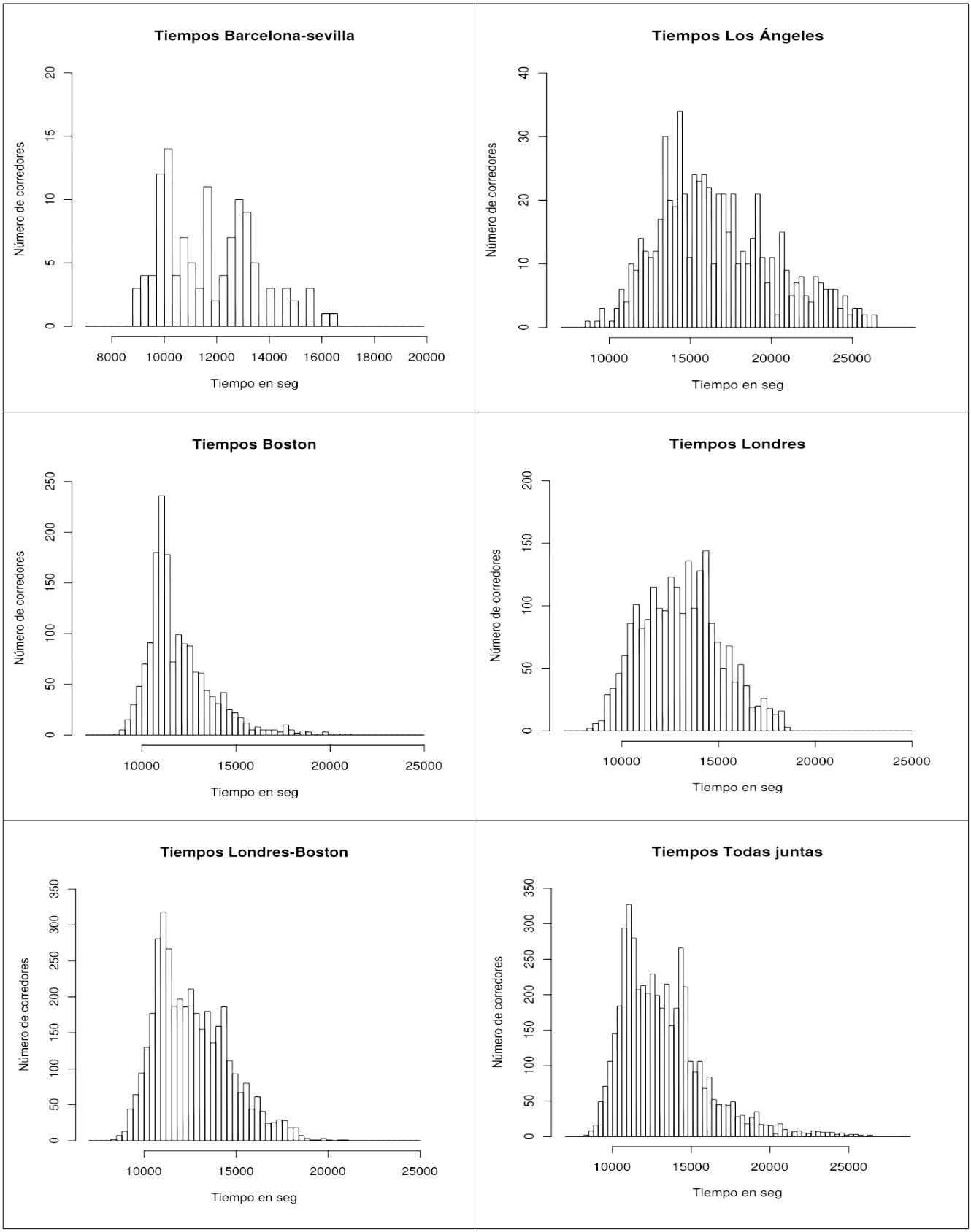

Figura 3: Distribuciones Tiempos Maratones Pre-Filtro

Otra información interesante se puede observar en las Figuras 4 y 5 donde se muestran el número de kilómetros a la semana por corredor. Ambas Figuras poseen la misma estructura, esto es, se componen de una tabla de 6 histogramas de los kilómetros totales por corredor realizados en cada semana. La nomenclatura utilizada nos indica la semana en la que se encuentra, siendo la semana 6 la semana previa a la maratón y la semana 1 la semana correspondiente a las 6 semanas previas a la prueba.

Ambas figuras muestran los datos para las maratones de Londres y Boston en conjunto. La Figura 4 corresponde a los datos antes del primer filtrado de datos. Lo que podemos observar en esta figura es que hay muchos corredores que tienen muy pocos kilómetros. Estos corredores se eliminarán tras el filtrado.

Si nos fijamos en los histogramas vemos que se desplazan hacia la derecha hasta la semana 4. Esto quiere decir que aumenta el número de kilómetros totales de los entrenamientos por semana. A partir de la semana 5 se comienza a desplazar hacia la izquierda, disminuyendo el número de kilómetros. Todo lo descrito es razonable ya que en las semanas 3 y 4 previas a la prueba es cuando se realiza un mayor número de kilómetros y con mayor intensidad. En estas semanas se logra un entrenamiento completo para en las dos últimas semanas, bajar el ritmo de entrenamiento para que el cuerpo asimile el esfuerzo previo y llegar a la prueba con el máximo estado de forma posible. Para consultar las gráficas del número de entrenamientos por semanas y tiempo total de los entrenamientos por semana basta consultar el **Anexo B**.

De igual manera en la Figura 5, donde se muestran los datos filtrados, vemos que el número de kilómetros por semana sube desde la semana 1 a la 4, y que a partir de la penúltima semana antes de la maratón el número de kilómetros decae en mayor grado.

A diferencia de la Figura 4, en esta vemos que el histograma no tiene el pico en los kilómetros cercanos a 0. Esto es gracias al filtro con el que se ha eliminado a todos los corredores que no tenían un mínimo de 3 entrenamientos por semana y que podían empeorar los resultados del estudio.

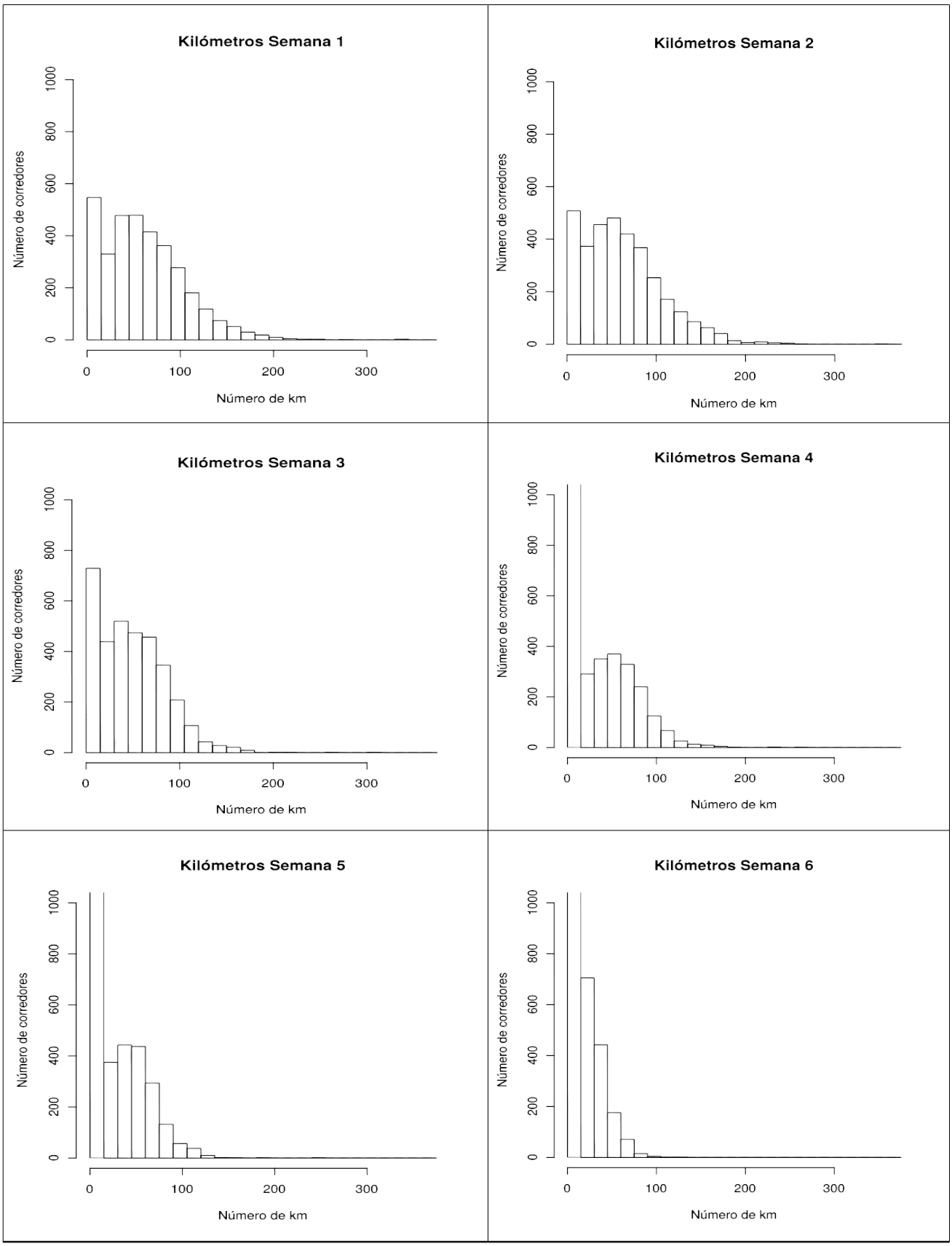

Figura 4: Comparativa gráficas kilómetros por semana en Londres y Boston Pre-Filtro

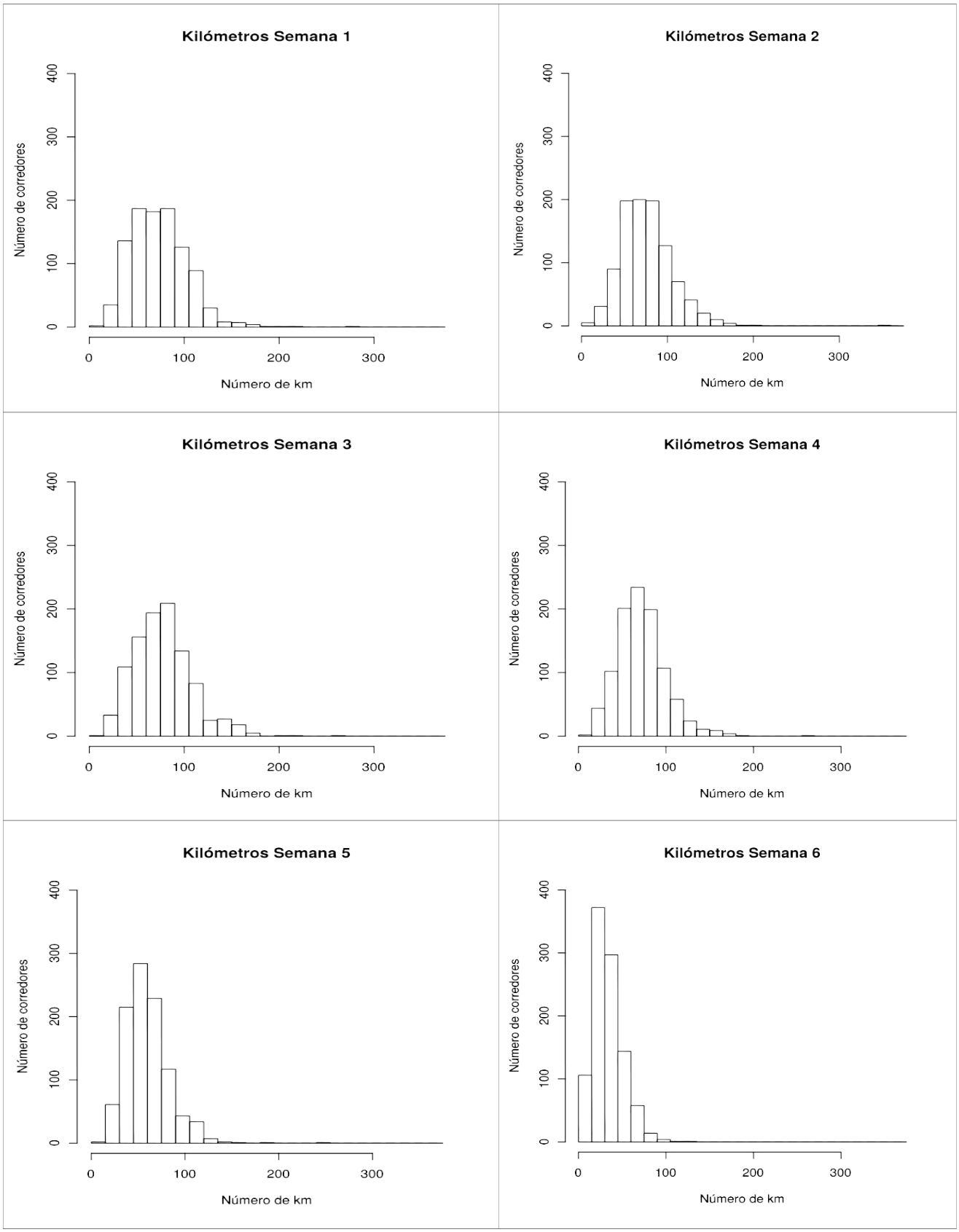

Figura 5: Comparativa gráficas kilómetros por semana en Londres y Boston Post-Filtro2

4 Pruebas y resultados

Las pruebas han sido realizadas con los algoritmos de aprendizaje automático de Weka. Los experimentos se han completado utilizando validación cruzada de 10 particiones. La validación cruzada consiste en dividir en n conjuntos disjuntos los datos de la muestra, y utilizar secuencialmente cada subconjunto como datos de test, y utilizando el resto de subconjuntos como datos de entrenamiento. Hay que repetir este procedimiento con cada subconjunto. El error utilizado es la media del error absoluto, ya que es el que se considera que tiene más interés que el error cuadrático a la hora de predecir un tiempo porque esta medida de error está en las mismas unidades que la variable objetivo. Para ver el significado de error absoluto consultar **2.2 Aprendizaje Automático**.

Con objeto de facilitar la reproducibilidad de los experimentos se dan las configuraciones y parámetros en los diferentes algoritmos de aprendizaje automático utilizados:

LinearRegression:

- AttributeSelectionMethod = M5 method
- EliminateConearAttributes = True
- Ridge = 10^{-8}

MultilayerPerceptron:

- AutoBuild= True
- Decay = False
- LearningRate = 0.3
- Momentum = 0.2
- Normalized Attributtes = True
- Training Time = 500
- ValidationSetSize = 0
- ValidationThreshold = 20

Bagging:

- BagSizePercent = 100,
- CalcOutBag = False
- Classifier = REPTree
- NumIterations = 10

IBK:

- KNN = 10,
- distance Weighting = Nodistance Weighting
- nearestNeigbourAlgo = LinearNNSearch con distancia euclídea

RandomSubSpace:

- classifier = REPTree sin poda,
- numIterations = 100,
- SubSpaceSize = 0.5

REPTree:

- maxDepth = -1,
- minNum = 2,
- minVariance = 0.001,
- noPrunning=False,
- numFolds = 3.

Zero R: no requiere parametrización. Este modelo, simplemente predice para todos los datos de test la media de la variable objetivo observada de los datos de entrenamiento. Este error es una buena medida para comprobar como de buena es la precisión del resto de algoritmos. Cualquier modelo que no pueda mejorar a ZeroR, no se considerará una buena aproximación. Consideraremos que un algoritmo tiene buenos rendimientos si la diferencia entre el error del algoritmo y de ZeroR es alta.

Es importante resaltar que para el caso del Perceptron los parámetros no han sido optimizados. Para este algoritmo se han usado los parámetros predeterminados por Weka salvo pequeñas variaciones. Este algoritmo se muestra por ser representativo.

Tiempos Máximos en las maratones

Una información relevante en el análisis de los resultados obtenidos es el tiempo máximo de finalización permitido por la organización de cada una de las maratones. Cuanto mayor es el tiempo máximo, mayores serán los rangos de predicción, lo que dificulta la tarea de predicción. Estos tiempos se pueden ver en la Tabla 4 y la Tabla 5. Estas tablas se componen de la prueba en cuestión y tiempo del corredor más lento del que tenemos entrenamientos. El tiempo mínimo en todas las maratones estaba cercano a las 2 h 20 minutos. Por este motivo no se muestra al no considerarse representativo. Hay que recordar que este tiempo es el tiempo mínimo de los corredores que han subido sus entrenamientos a la plataforma y no el tiempo del ganador de la prueba.

Pre-Filtro

Carrera	Tiempo maximo
Los Ángeles	7h 20 min
Boston	5h 47 min
Londres	5h 4 min
Barcelona y Sevilla	4h 34 min

Tabla 4: Tiempo máximo maratones Pre-Filtro

Post-Filtro

Carrera	Tiempo maximo
Los Ángeles	5h 44 min
Boston	4h 59 min
Londres	4h 05 min
Barcelona y Sevilla	3h 01 min

Tabla 5: Tiempo máximo maratones Post-Filtro

4.1 Resultados Pre-Filtro

Los resultados de la clasificación se muestran en la Tabla 6. Esta tabla está estructurada de la siguiente manera. En la primera columna tenemos los clasificadores y en las siguientes el error de cada clasificador en cada una de las estructuras temporales. En esta tabla vemos los errores absolutos medios en segundos correspondientes a los datos de Barcelona y Sevilla conjuntamente. Se utilizan estas dos maratones en conjunto debido al bajo número corredores disponibles por las razones analizadas anteriormente. Esta tabla muestra el error absoluto medio para 12 modelos distintos.

Clasificador	Temporalidad: Semanal	Temporalidad: 2semanas	Temporalidad: Aglomerativo
GaussianProcess	1223,34	964,81	1334,27
LinearRegression	1355,23	960,30	1040,89
MultilayerPerceptron	1427,59	1549,98	1518,44
Bagging	826,97	747,70	698,15
DecisionTable	1031,60	1084,29	898,76
m5Rules	1117,06	1204,05	888,85
m5P	836,14	899,90	844,77
REPTree	947,12	939,69	892,32
IBK	1370,68	1005,91	1307,28
DecissionStump	981,88	956,40	972,54
RandomSubspace	831,56	743,02	714,43
ZeroR	1488.73	1461.36	1485.51

Tabla 6: Error medio absoluto(segundos) Barcelona-Sevilla Pre-Filtro

A la vista de los resultados de la Tabla 6, vemos que Bagging y RandomSubSpace sobresalen por encima del resto reduciendo prácticamente a la mitad el error de ZeroR. Para el resto de pruebas del capítulo se eligen éstos algoritmos junto con algún otro de los más representativos. Se asume que por la similitud del formato de los datos y del problema, estos algoritmos

seguirán funcionando mejor que el resto. Además, en esta tabla podemos observar que los mejores resultados siempre se dan para el caso de temporalidad aglomerativa. Esto puede ser normal ya que los atributos del formato aglometarivo incluyen todos los atributos del formato semanal y bisemanal. Se opta también por hacer las pruebas siguientes solamente para este tipo de formato.

A partir de este momento las tablas seguirán el siguiente formato. Las tablas se componen de una columna en la que aparece el clasificador y en el resto de columnas encontramos el error medio absoluto en segundos del clasificador con el conjunto de datos de esa maratón. En todos los casos con temporalidad será aglomerativa.

En la Tabla 7 encontramos los datos de las maratones antes de aplicar el filtro. Observamos que el mejor algoritmo vuelve a ser Bagging para todas las maratones. Merece especial mención la agrupación de datos de Londres y Boston. Se han unido los datos de estas maratones puesto que son las maratones más similares y en la que mejores resultados de predicción se obtienen. De esta manera tenemos un conjunto de datos mayor.

Clasificador	Londres	Boston	Londres-Boston	Los Ángeles	Todos
LinearRegression	873.0	886.4	863.1	1909.5	1139.9
MultilayerPerceptron	1069.9	1200.1	946.8	3140.2	1276.3
Bagging	699.1	712.9	707.8	1688.1	942.6
IBK	1132.1	1026.1	923.9	2890.9	1453.5
RandomSubspace	704.3	728.7	720.3	1704.6	946.1
REPTree	778.5	835.4	812.3	1801.4	1088.8
ZeroR	1380.5	1331.5	1335.6	2930.6	1877.5

Tabla 7: Error absoluto medio(segundos) Pre-Filtro

Las sucesivas figuras mantendrán el siguiente formato. Estas figuras se componen de varios histogramas que representan la dispersión del error del algoritmo de Bagging con una muestra de test del 30%. El eje de coordenadas representa el tiempo real, mientras que el eje de ordenadas representa el tiempo predicho.

En la Figura 6 podemos observar los errores de las distintas maratones. En el caso de Barcelona-Sevilla, debido a la baja muestra no podemos apreciar el error de forma clara. Destaca la dispersión de la gráfica de todas las maratones juntas. Esta dispersión se acentúa en los tiempos superiores a 18000 segundos, esto es 5 horas,. También es destacable la gran masa de puntos con baja dispersión de los datos de Londres-Boston conjuntas.

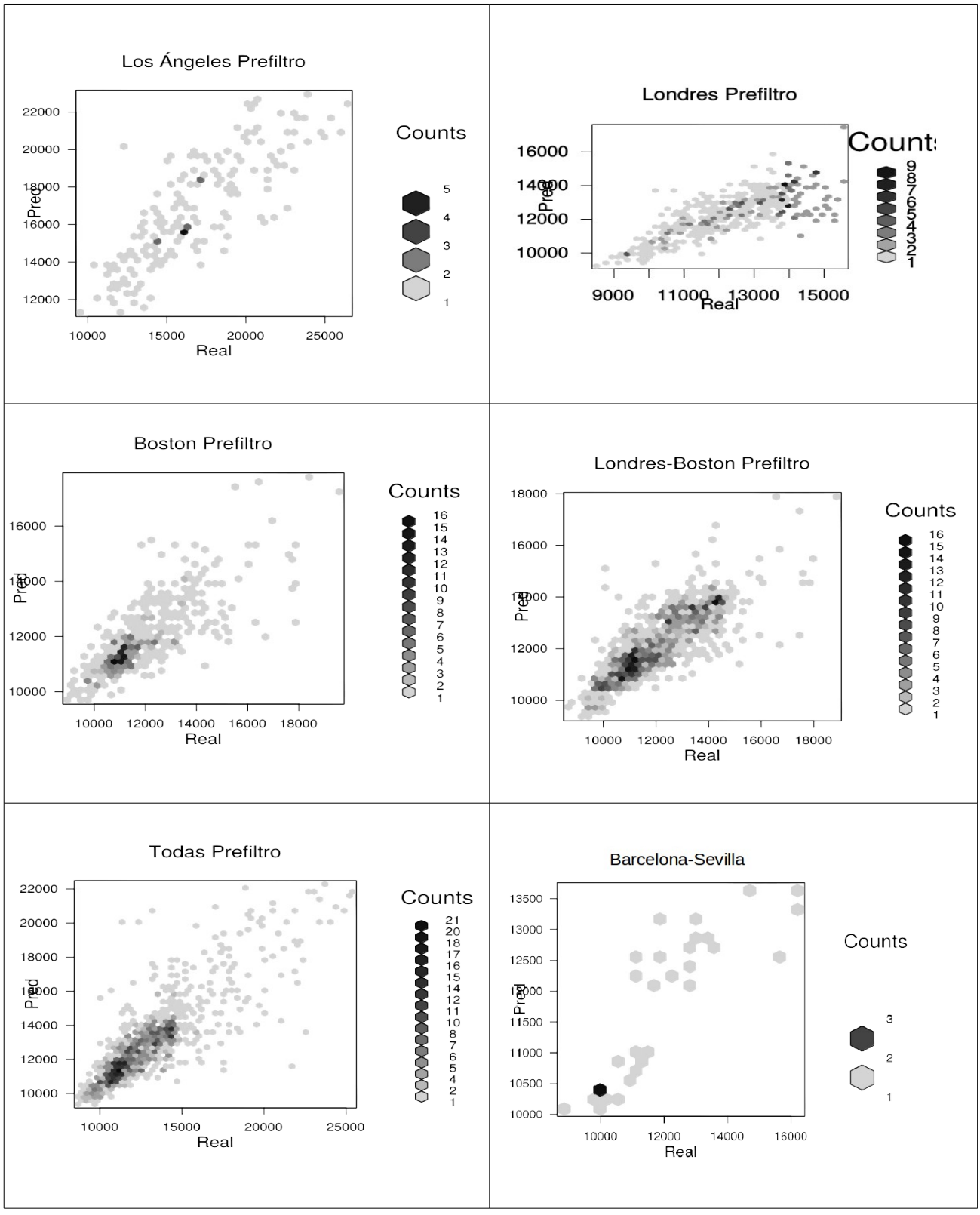

Figura 6: Hexplot de dispersión de errores Pre-Filtro

Por último hay que fijarse en la banda de dispersión que se tiene en los resultados de Los Ángeles. Esto provoca que el error medio sea más alto que para el resto de carreras. Los resultados menos precisos de la maratón de Los Ángeles son debidos a que el tiempo máximo en esta maratón es mucho mayor que en el resto, por lo que es comprensible que los errores sean mayores. Sólo hace falta observar el error de ZeroR para comprobar que al ser éste mucho mayor va a ser más complicado mejorar tanto como para alcanzar los niveles de error de las otras maratones.

4.2 Resultados Post-Filtro

Tras el análisis de los datos se detectó un número muy bajo de entrenamientos en algunas semanas de algunos corredores. Se dedujo que este podía ser un motivo que limitara la precisión de la regresión. La hipótesis seguida es que los corredores con pocos entrenamientos podrían tener un vector de atributos similar al de un corredor de menor nivel que suba todos los entrenamientos realizados. Es esto caso, estos corredores se aproximarían mal por la regresión. Por este motivo se procedió al filtrado de los corredores con menos tres entrenamientos en alguna de las semanas del periodo considerado, excepto para la semana previa a la competición.

Los resultados después del filtrado de corredores con menos de 3 entrenamientos en alguna semana se pueden observar en la Tabla 8 para Barcelona y Sevilla y en la Tabla 9 para el resto de maratones.

En la Tabla 8 observamos que el mejor algoritmo sigue siendo Bagging. Es destacable que el filtrado en este conjunto de datos no consigue un resultado mejor que el conjunto original posiblemente debido al bajo número de corredores. En la tabla 9 vemos una mejora para casi todos los algoritmos utilizados. Fijándonos específicamente en el algoritmo con mejores resultados, Bagging, vemos que en Londres-Boston pasamos de 707 a 569, esto es una mejora del 20%. En el caso de Los Ángeles seguimos teniendo unos resultados no tan buenos como para el resto de maratones. Sin embargo se observan unas mejoras de un 25% respecto a los datos originales de esta maratón.

Clasificador	Temporalidad: Aglomerativo
LinearRegression	751.01
MultilayerPerceptron	905.19
Bagging	706.57
IBK	1052.80
RandomSubspace	717.47
REPTree	778.78
ZeroR	983.33

Tabla 8: Error medio absoluto(segundos) Barcelona y Sevilla Post-Filtro

Clasificador	Londres	Boston	Londres-Boston	Los Ángeles	Todos
LinearRegression	762,1	712,1	663,7	1769,2	786,9
MultilayerPerceptron	939,2	970,8	845,2	2905,8	921,1
Bagging	564,3	582,9	569,0	1208,1	639,5
IBK	806,8	757,5	801,6	1741,3	912,3
RandomSubspace	581,8	577,7	571,8	1177,1	649,4
REPTree	784,5	751,4	721,9	1461,3	792,6
ZeroR	920,1	1019,3	1057,3	1769,2	1222,0

Tabla 9: Resultados clasificadores Post-Filtro

En la Figura 7 podemos observar la dispersión de los errores de los conjuntos de datos Post-Filtro. En los casos de Barcelona-Sevilla y Los Ángeles debido a la baja muestra no se ha podido representar el error de forma gráfica. La principal diferencia respecto a las gráficas Pre-Filtro es la menor dispersión que observamos en los errores. Esto se visualiza especialmente bien en el caso de Londres y Boston fusionadas.

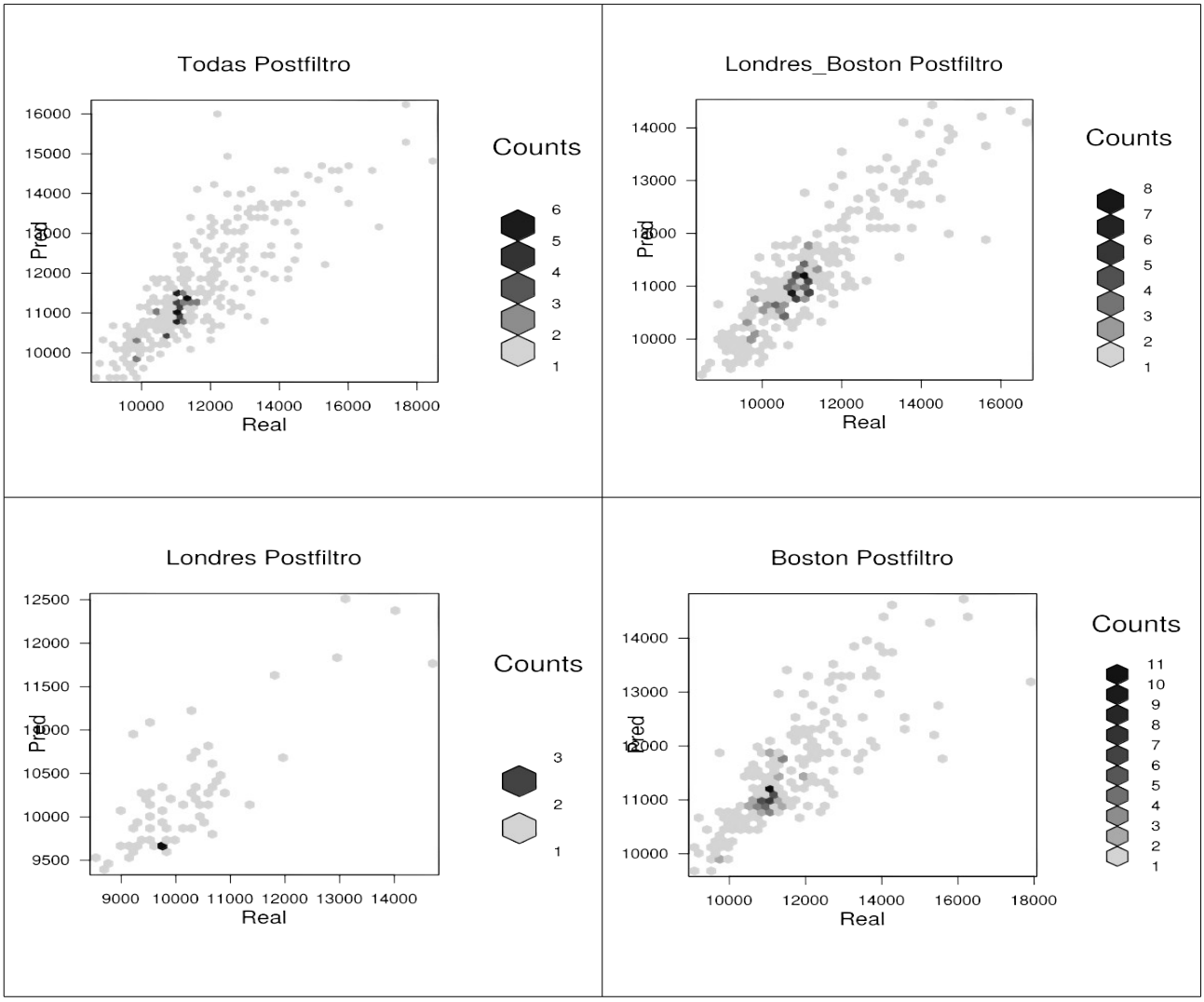

Figura 7: Hexplot dispersión de errores Post-Filtro

Posteriormente se hizo un último filtrado para las maratones Londres-Boston por ser las pruebas con mejores resultados. Esta criba se corresponde con **3.3 Fase 3:Filtrado** segunda etapa. Debido a la existencia de algún dato atípico, como son entrenamientos de más de 80 kilómetros, se procedió a eliminar esos entrenamientos. Específicamente los filtros fueron: número kilómetros de un entrenamiento mayores a 50 y ritmos de entrenamiento menores de 120 s, es decir ritmos de menos de 2min/km. Una comparativa para corroborar el razonamiento es que el récord del mundo de 1 kilometro, que ostenta el keniata Noah Ngeny es 2:11.96 por lo que no parece razonable que haya entrenamientos con esos ritmos. Los resultados de este filtro pueden observarse en la Tabla 10. En este caso el error en Bagging baja hasta los 555 segundos. Enlazando este error con la Figura 8 vemos la dispersión del error para este conjunto y comprobamos que es muy similar al anterior.

Londres Boston

Clasificador	Temporalidad: Aglomerativo
LinearRegression	658.03
MultilayerPerceptron	2052.81
Bagging	555.43
IBK	692.39
RandomSubspace	563.12
REPTree	685.15
ZeroR	1054.66

Tabla 10: Error medio absoluto(segundos) Londres y Boston Post-Filtro2

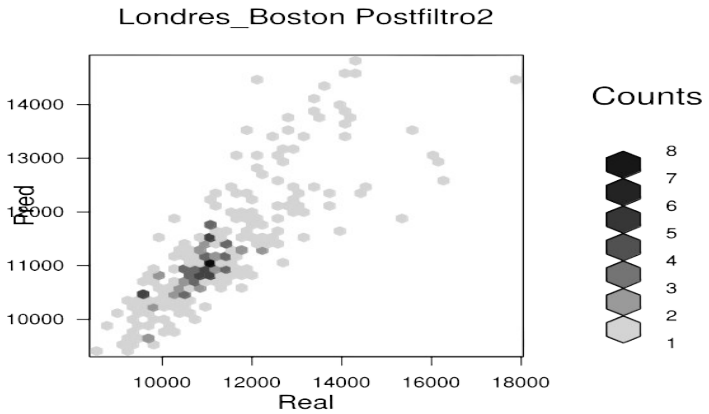

Figura 8: Hexplot de dispersión de errores Londres-Boston Post-Filtro2

Finalmente, haciendo uso de Knime con los algoritmos de Weka se realizó un filtrado simulando los corredores que tenían una predicción mucho menor que la marca real. Esto se relaciona con corredores que pudieron tener algún problema durante la carrera, como alguna lesión, o falta de nutrientes, reproduciendo lo que en argot de running se conoce como "una pájara". En esta fase del proceso descartamos aquellos usuarios que tenían una predicción de 900 segundos más rápidos que el tiempo real en la carrera.

Los resultados de la Tabla 11 son los correspondientes a ésta última criba, que se corresponde con los corredores que pudieron sufrir un problema en carrera. En este caso obtenemos mediante Bagging un error de 436 para los corredores que no experimenten percances serios. Esto corresponde con una mejora de un 20% respecto del anterior filtro.

Londres Boston

Clasificador	Temporalidad: Aglomerativo
LinearRegression	530.4291
MultilayerPerceptron	848.8649
Bagging	436.2675
IBK	587.9034
RandomSubspace	463.1884
REPTree	557.3184
ZeroR	900.5758

Tabla 11: Error medio absoluto(segundos) Londres y Boston Filtro-Lesiones

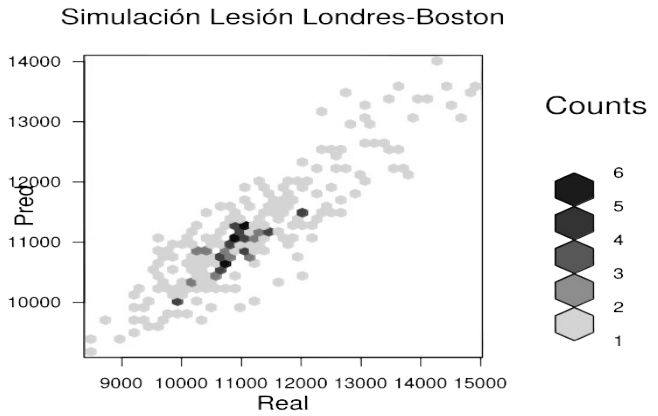

Figura 9: Hexplot de dispersión de errores Simulación lesión Londres-Boston

En la Figura 9 observamos la gráfica de dispersión del error de este conjunto de datos sin lesiones. Como podemos observar la dispersión del error es aún más baja que en los casos anteriores. Por este motivo el error es menor. Con este filtrado llegamos al conjunto de datos con menor error para la clasificación del tiempo de una maratón.

Para comprobar que todas las cribas que hemos realizado tienen sentido y que no nos hemos limitado a eliminar de nuestro conjunto de datos los corredores que tienen un mayor error se ha realizado la siguiente comprobación. Se han cargado los datos con los entrenamientos de Londres-Boston originales sin criba y los correspondientes a este último filtrado. Llamaremos D1 al primero y D2 al segundo. En primer lugar se ha escogido una muestra de D2 del 30% que serán el test y que llamaremos D2Test. A continuación se ha eliminado de D1 y de D2 los datos de test extraídos de D2, esto es D2Test. A estos dos conjuntos de datos los llamaremos D1Train y D2Train. Con estos dos conjuntos se ha entrenado el algoritmo de Bagging. Por último se ha calculado el error absoluto medio en D2Test para ambos conjuntos de Bagging. Los resultados obtenidos son de 520 segundos cuando se entrena con D2Train mientras que cuando el bagging ha sido entrenado con D1Train el error es de 580 segundos, a pesar de que D1Train tiene más datos que D2Train. Esto significa que el filtrado seguido sí que mejora realmente la predicción.

4.3 Datos externos

Finalmente se consiguió que un atleta amateur de maratones nos cediera sus entrenamientos de las 6 semanas anteriores a la maratón de Vitoria del 15 de Mayo. Esta prueba no es estadísticamente relevante por tratarse de un sólo sujeto, pero nos sirve como prueba de concepto para ejemplificar cómo podrían aplicarse los modelos obtenidos.

La Tabla 12 nos muestra la fecha, distancia, tiempo, ritmo y desnivel de los entrenamientos del atleta durante las 6 semanas previas a la prueba. Estos entrenamientos se han procesado con un programa similar al usado en **3.4 Limpieza y Formateo de los Datos**. Mediante este programa Java hemos dado formato Arff a estos entrenamientos en la temporalidad aglomerativo como hemos estado haciendo en los anteriores procesados.

Fecha	Distancia	Tiempo	Ritmo	Desnivel
03/04	21.35	01:49:29	05:08	319
05/04	20.42	01:50:35	05:20	92
07/04	16.55	01:28:11	05:15	83
10/04	20.75	01:49:02	05:15	305
14/04	16.55	01:27:52	05:20	121
17/04	23.72	02:21:07	05:57	552
19/04	9.92	00:53:38	05:25	65
21/04	16.60	01:28:03	05:18	96
24/04	32.47	02:52:12	05:18	615
26/04	9.75	00:57:38	05:48	23
28/04	14.60	01:21:32	05:38	64
30/04	7.75	00:42:45	05:31	55
02/05	24.64	02:15:29	05:29	212
07/05	16.28	01:32:31	05:41	123
12/05	9.95	00:54:23	05:28	51

Tabla 12: Entrenamientos Pre-Maratón de Ángel

El atleta es un varón de 52 años. El resultado que consiguió el atleta en su prueba de Vitoria se puede contemplar en la Figura 5. El tiempo que nos interesa es el tiempo real, es decir, 4 horas 4minutos, o lo que es lo mismo 14684 segundos.

La predicción del atleta con el algoritmo de Bagging y los distintos conjuntos de entrenamiento pueden verse en la Tabla 17. En esta tabla apreciamos que el error es mayor que la media de los errores obtenidos en los anteriores resultados.

Cabe resaltar las malas sensaciones del corredor en los últimos 7 kilómetros. En este tramo final el ritmo que llevaba hasta ese momento de 5:18 minutos/kilómetro se disparó hasta los 7:00 min/km. Esto se debe según nos cuenta el corredor a su donación de sangre por motivos relevantes en días previos a la maratón. No obstante, si el corredor no hubiera tenido percances durante la prueba y hubiera continuado a un ritmo de 5:18, salvo el pequeño bajón que sufren la casi totalidad de corredores en los últimos kilómetros de un maratón, se habría obtenido una predicción razonable. La subida en el ritmo por kilómetro en los últimos kilómetros de un maratón es de un 10% aproximadamente. En nuestro caso este 10% le habría llevado a un ritmo de unos 5:45 en los últimos 6 kilómetros. De esta manera su marca habría sido de unos 11448+2040 segundos, es decir de 13518 segundos, que traducido en horas son 3h 45 minutos. Por lo tanto si no hubiera tenido problemas estimamos que la predicción dada por el sistema propuesto es más que razonable con una desviación en torno a los 5 minutos.

Pos.	Zbk.	Corredor	T. oficial	Dif.	T. Real	Ritmo	P.	Km 21	Provincia	P.	Categoria
691	704	[redacted] ANGEL JAVIER	4 :06:00	1:48:16	4:04:44	5:48	553	0:27:11	MADRID	97	VET D Mas

Figura 10: Resultado maratón de Vitoria de Ángel

Conjunto de Datos	Predicción	Error
Londres-Boston Prefiltro	13822	862
Londres-Boston Postfiltro	13717	967
Londres-Boston Postfiltro2	13857	827
Londres-Boston Sin-Lesión	13818	866

Tabla 13: Error Maratón Vitoria

5 Conclusiones y trabajo futuro

5.1 Conclusiones

En este trabajo se ha propuesto el uso de técnicas de aprendizaje automático para la predicción de tiempos en carreras de fondo para corredores populares a partir de datos de entrenamientos. Para alcanzar este objetivo el trabajo se ha estructurado en dos fases principales. Primero ha sido necesaria la obtención de los datos de entrenamientos de un gran número de corredores populares mediante técnicas de scrapping. La segunda fase ha consistido en el análisis de los datos utilizando algoritmos de aprendizaje automático.

Para la primera fase del trabajo se ha tenido que buscar una fuente de datos fiable de entrenamientos y de la que se pudiera recuperar la información. La página y aplicación de Strava. Esta red social deportiva permite a los usuarios subir sus entrenamientos de multitud de deportes. Para extraer los datos de las marcas y entrenamientos de los corredores de forma automática se han usado Wget y CasperJS. Este último nos ha permitido ejecutar Javascript y automatizar completamente el proceso. En total se han extraído 119.960 entrenamientos de 4128 corredores de las maratones de Sevilla, Barcelona, Los Ángeles, Boston y Londres.

Además en esta fase ha sido imprescindible comprobar la consistencia e integridad de los datos para que se pudieran obtener predicciones válidas. Se ha realizado un completo procesado de los datos en crudo para excluir de nuestro conjunto de datos registros incoherentes, inválidos, o anómalos. Ha sido crucial eliminar los entrenamientos en bicicleta de los usuarios, así como la exclusión de aquellos usuarios que, o bien subían menos entrenamientos de los que realmente realizaban, o bien subían entrenamientos en otras aplicaciones o plataformas. Hemos observado que incluir los datos de estos corredores con información incompleta desembocaba en mayores errores en la regresión.

En la segunda fase de este trabajo se ha realizado un análisis de los datos obtenidos mediante algoritmos de aprendizaje automático. Para poder aplicar estos algoritmos primero se han transformado los datos, extrayendo los atributos relevantes como distancia, tiempo, ritmo y desnivel. Se han probado multitud de algoritmos con los diferentes conjuntos de datos. El algoritmo que mejor ha funcionado ha sido Bagging con el conjunto de atributos aglomerativo. En concreto podemos afirmar que con nuestro conjunto de datos de Londres-Boston, sin los corredores que sufrieron percances durante la carrera, y el algoritmo de Bagging, se consigue aproximar su marca con un error medio de 436 segundos.

Finalmente se ha realizado una prueba de concepto de uso de las técnicas desarrolladas. Hemos trabajado con los datos de un atleta que realizó la maratón de Vitoria del 15 de Mayo de 2016. Los buenos resultados obtenidos nos hacen ver la potencia del análisis de datos y cómo con un muestreo de webs de running podemos conseguir aproximar los tiempos de una maratón.

La conclusión final de este proyecto es que se puede usar la información de los entrenamientos para hacer predicciones precisas de los tiempos de maratón. Es importante resaltar que esto ha sido en parte posible gracias a la potencia de las técnicas de scrapping actuales que nos han permitido descargar una gran cantidad de datos de la web.

5.2 Trabajo futuro

Para continuar el trabajo realizado, sería de gran ayuda contar con un mayor número de usuarios y de entrenamientos para mejorar los resultados obtenidos, así como la posibilidad de usar datos de entrenamientos para predecir marcas de carreras de 10 kilómetros o medias maratones. De igual manera sería interesante poder predecir los tiempos de un Triatlon o un Ironman.

Dado el volumen de descargas, sería enormemente útil contar con un servidor con mayor velocidad de descarga y un ordenador con más potencia. Además para este proyecto en el que la cantidad de datos mejora los algoritmos sería interesante trabajar con tecnologías Big Data.

Otro posible trabajo relacionado con este proyecto consiste en analizar los datos de los ritmos cardíacos de los corredores y tratar de predecir los tiempos mediante su capacidad cardiaca, así como prevenir posibles insuficiencias o problemas cardiacos.

Como última idea para continuar con este trabajo se podría crear una aplicación con una interfaz gráfica agradable y que fuera compatible con las diversas plataformas online en las que los corredores incorporan sus entrenamientos. Esta aplicación recogería la información de la maratón que se pretende realizar y haría una predicción del tiempo en la carrera con los entrenamientos previos. Esto serviría para comprobar como llevamos los entrenamientos y si tenemos que aumentar la intensidad de los mismos para alcanzar nuestras metas.

Referencias

[1] Mohammed J Zaki and Wagner Meira Jr. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, 2014.

[2] David J Hand, Heikki Mannila, and Padhraic Smyth. Principles of data mining. MIT press, 2001.

[3] What is a Headless Browser. http://blog.arhg.net/2009/10/what-is-headless-browser.html

[4] Pattern Recognition (fourth edition). S. Theodoridis and K. Koutroumbas. Academic Press, 2009.

[5] J. Ross Quinlan. C4.5: Programs form Machine Learning, 1993.

[6] Tin Kam Ho. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell., 20(8):832–844, 1998. d o i : 10.1109/34.709601.

[7] https://runningdv.wordpress.com/2014/03/30/estadisticas-de-la-participacion-en-maratones-en-espana-2008-2013/

[8] http://www.europapress.es/deportes/noticia-grupos-running-claves-auge-deporte-espana-20141213112752.html

[9] Bagging Predictors By Leo Breiman Technical Report No. 421 September 1994 Department of Statistics University of California Berkeley, California 94720

[10] http://casperjs.org/

[11] http://phantomjs.org/

[12] http://www.runedia.com/curses/pais/Espa%C3%B1a/ccaa/Catalunya/

[13] http://www.bbc.com/mundo/noticias/2015/05/150514_deportes_pionera_maraton_kathrine_switzer_ng_finde

Glosario

API	Application Programming Interface
ARFF	Attribute Relation File Format
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPs	HyperText Transfer Protocol Secure
SQL	Standard Query Language
URL	Uniform Resource Locator
ETL	Extract, Transform and Load

Anexos

A Scripts

A.1 Script Bash

```
#!/bin/bash

# -*- ENCODING: UTF-8 -*-

I=0

J=1

POSHTML=1

ID=0

mkdir "ENTRENOS"

while read line

do

   let I+=1

   IN=$line

arrIN=(${IN//,/ })

   NOMBRE=$( echo ${arrIN[0]}  )

   SEXO=$( echo ${arrIN[3]}  )

   EDAD=$( echo ${arrIN[4]}  )

   TIEMPO=$( echo ${arrIN[2]}  )

   RITMO=$( echo ${arrIN[1]}  )

   ENTRENAMIENTOS=$(grep -A2 "<div class=\"app-icon icon-run icon-sm type\" title=\"Carrera\"></div>" $
{POSHTML}.html |grep -E --only-matching 'href=".+"[ |>]'| cut -c 7-49  | cut -d "\"" -f 1 | sed
's/\/ac/https:\/\/www.strava.com\/ac/g'  )

   echo -e $ENTRENAMIENTOS > temp.txt

   sed 's/ /\n/g' temp.txt > temporal

   let POSHTML+=1

   let ID+=1
```

```
while read linea

do

    wget --load-cookies=cookies.txt $linea -O ENTRENOS/${J}.txt

    NAME=$(grep "<a href=\"/athletes/" ENTRENOS/${J}.txt | grep "class=\"minimal" | grep -e ">.*<" --only-matching | sed 's/<//' |sed 's/>//')

    FECHA=$(grep -A1 "<time>" ENTRENOS/${J}.txt| grep -e ', ' | cut -d " " -f 2,3,4)

    ATRIBUTOS=$(grep "<strong>" ENTRENOS/${J}.txt | head -n 4 | sed 's/<[a-z]*>//g' | sed 's/<\/abbr><\/strong>\/km//g' | sed 's/<.*>//g')

    echo $ATRIBUTOS > temp.txt

    ATRIBUTOS=$(sed 's/ /;/g' temp.txt )

    #meto en una linea NOMBRE FECHA ATRIBUTOS

    echo -e $FECHA \; $ATRIBUTOS \; $SEXO \; $EDAD \; $TIEMPO \; $RITMO \; $ID >> DATOS.TXT

    let J=J+1

done < temporal

ENTRENAMIENTOS=$(grep -A2 "<div class=\"app-icon icon-run icon-sm type\" title=\"Carrera\"></div>" ${POSHTML}.html |grep -E --only-matching 'href=".+"[ |>]'| cut -c 7-49 | cut -d "\"" -f 1 | sed 's/\/ac/https:\/\/www.strava.com\/ac/g' )

echo -e $ENTRENAMIENTOS > temp.txt

sed 's/ /\n/g' temp.txt > temporal

let POSHTML+=1

while read linea

do

    wget --load-cookies=cookies.txt $linea -O ENTRENOS/${J}.txt

    NAME=$(grep "<a href=\"/athletes/" ENTRENOS/${J}.txt | grep "class=\"minimal" | grep -e ">.*<" --only-matching | sed 's/<//' |sed 's/>//')

    FECHA=$(grep -A1 "<time>" ENTRENOS/${J}.txt| grep -e ', ' | cut -d " " -f 2,3,4)

    ATRIBUTOS=$(grep "<strong>" ENTRENOS/${J}.txt | head -n 4 | sed 's/<[a-z]*>//g' | sed 's/<\/abbr><\/strong>\/km//g' | sed 's/<.*>//g')

    echo $ATRIBUTOS > temp.txt

    ATRIBUTOS=$(sed 's/ /;/g' temp.txt )

   #meto en una linea NOMBRE FECHA ATRIBUTOS
```

```
        echo -e $FECHA \; $ATRIBUTOS \; $SEXO \; $EDAD \; $TIEMPO \; $RITMO \; $ID >> DATOS.TXT
        let J=J+1
    done < temporal
done < Boston_faltaba.csv
exit
```

A.2 Script CasperJ

```
var casper = require('casper').create();
var cookieFileName = 'cookie.txt';
casper.userAgent('Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)');
phantom.cookiesEnabled = true;
casper.echo('Cookies enabled?: ' + phantom.cookiesEnabled);
casper.start('https://www.strava.com/login', function() {
   casper.echo("carga");
 });
casper.then(function(){
//Hacemos el login
        casper.fill('form[action="/session"]',{
                email: 'user@gmail.com',
                password: 'Prueba1'
                }, true);
});
casper.then(function(){
});
// grab cookies from file
var fs = require('fs');
var utils = require('utils');
var cookies = JSON.stringify(phantom.cookies);
fs.write(cookieFileName, cookies, 644);
casper.page.setCookies(cookies);
casper.then(function readFile() {
        stream = fs.open('ang_cor.csv', 'r');
        line = stream.readLine();
        i = 1;
        while(line) {
                //casper.echo(line);
        casper.thenOpen(line, function()
        {       casper.echo(i);
        });
        casper.wait(10000);
        casper.then(function(){
```

```
                    var aux= i+".html" ;
                    fs.write(aux, this.getHTML(), 644);
                    i++;
            });
                    line = stream.readLine();
            }
});
//casper.page.setCookies(cookies); y pondria las cookies
casper.run();
```

B Gráficas

Gráficas correspondientes a los corredores de maratón de Londres y Boston juntos.

En primer lugar, en las Figuras 11 y 12 tenemos la secuencia del número de entrenamientos antes del filtrado por corredor. Estas figuras se componen de 6 histogramas que nos dan información del número de corredores que ha realizado un determinado número de entrenamientos. La nomenclatura seguida es la explicada en anteriores secciones, recordamos que la semana 6 es la semana previa a la maratón.

En la Figura 11 observamos cómo muchos corredores tienen 0 entrenamientos y es por este motivo por el que se realizó el filtro de la **3.3 Fase 3: Filtrado**.

En la Figuras 13 y 14 observamos los tiempos totales por semanas de los atletas de las maratones de Londres y Boston antes y después del filtro **3.3 Fase 3: Filtrado**. Estas figuras se componen de 6 histogramas en las que se muestra el número de corredores por tiempos totales de entrenamientos semanales.

Los tiempos están en horas. De igual manera que en la Figura 11 en la Figura 13 tenemos un número de tiempos muy cercanos a cero. Estos entrenamientos de escasa duración se reducen gracias al filtro. Sin embargo en las Figuras 12 y 14 no vemos los picos cercanos a 0, gracias a la criba de corredores.

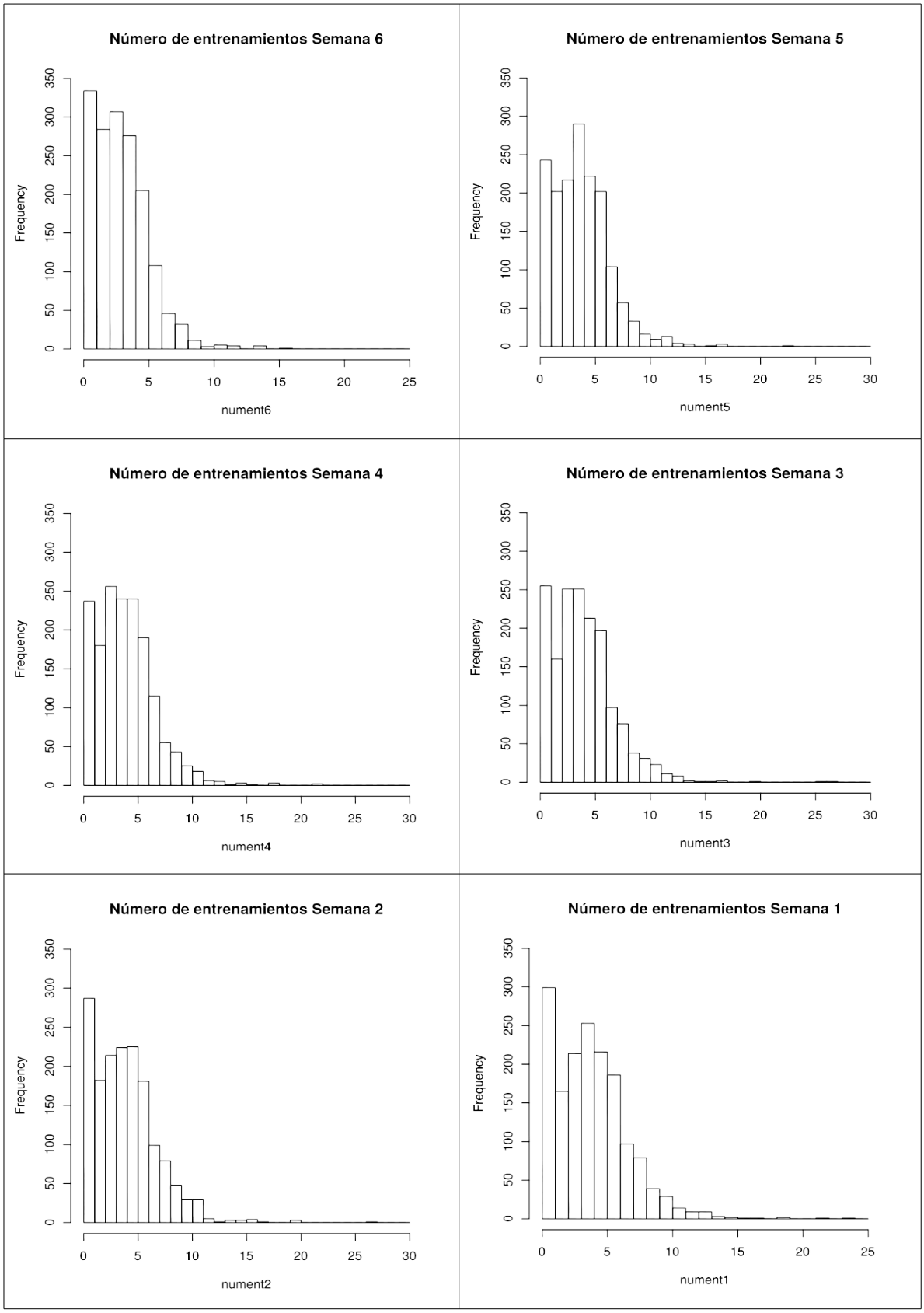

Figura 11: Distribuciones número de entrenamientos por corredor en Londres-Boston Pre-Filtro

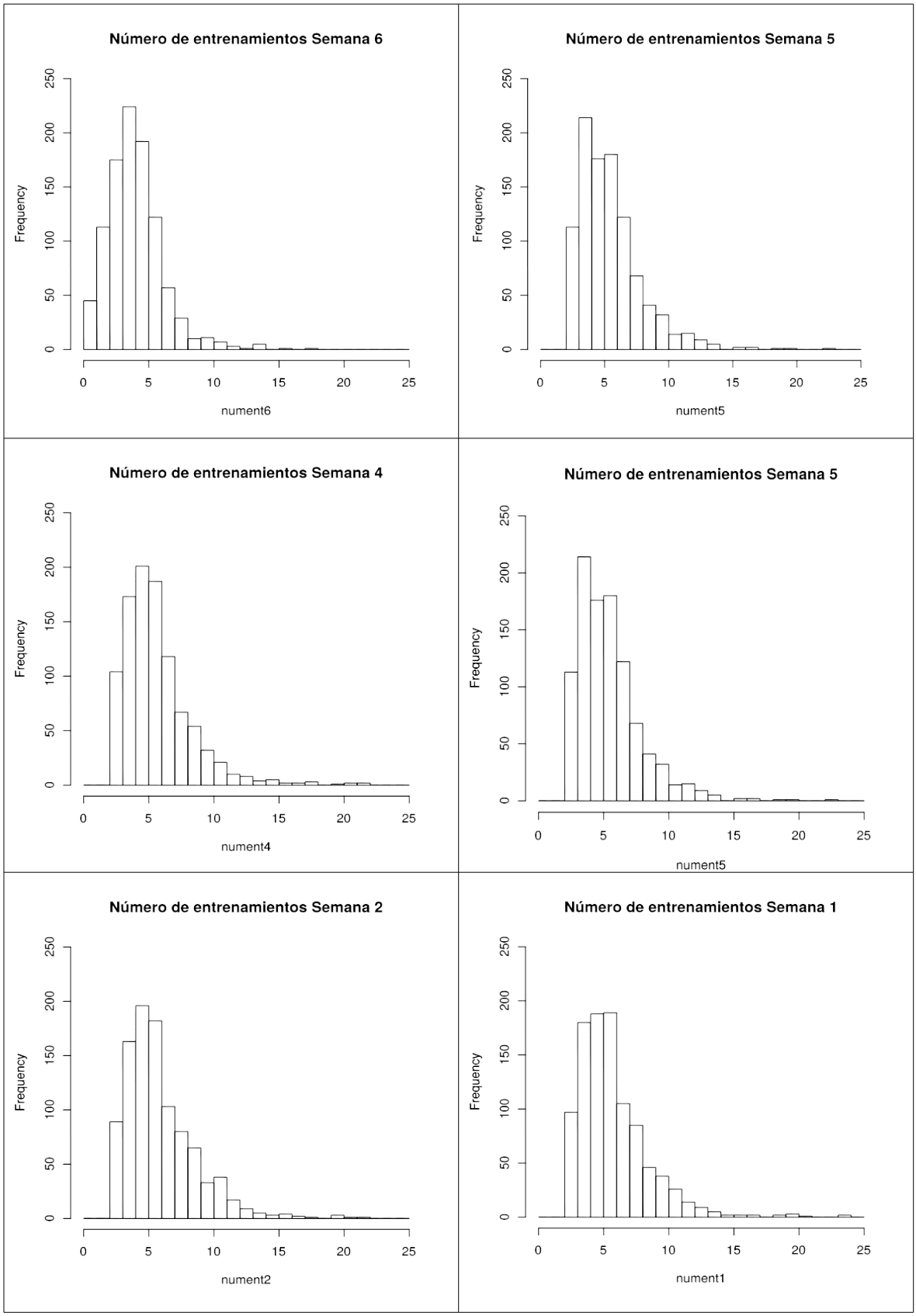

Figura 12: Distribuciones número de entrenamientos por corredor en Londres-Boston Post-Filtro

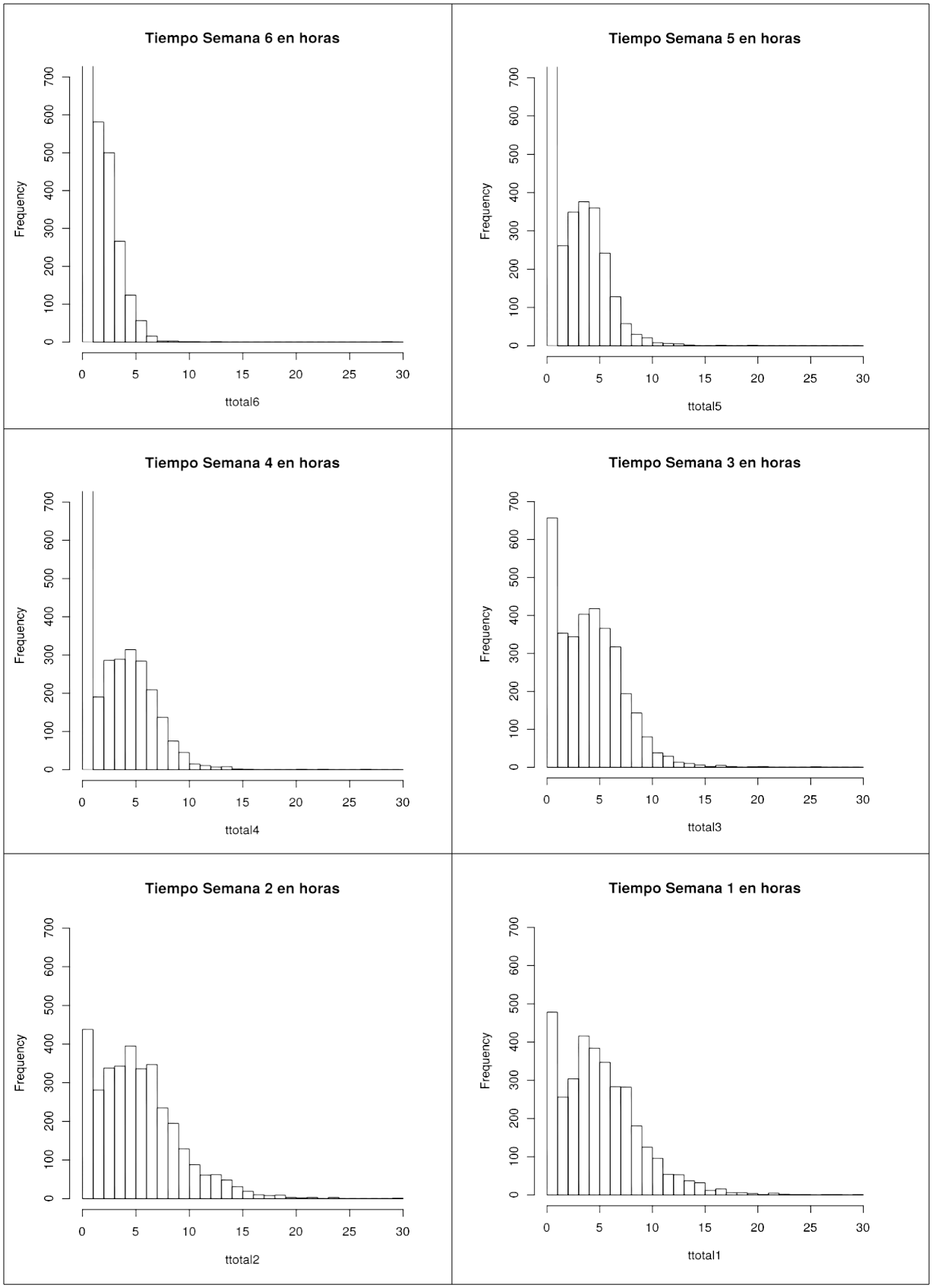

Figura 13: Tiempos totales de entrenamientos por corredor por semana Pre-Filtro Londres-Boston

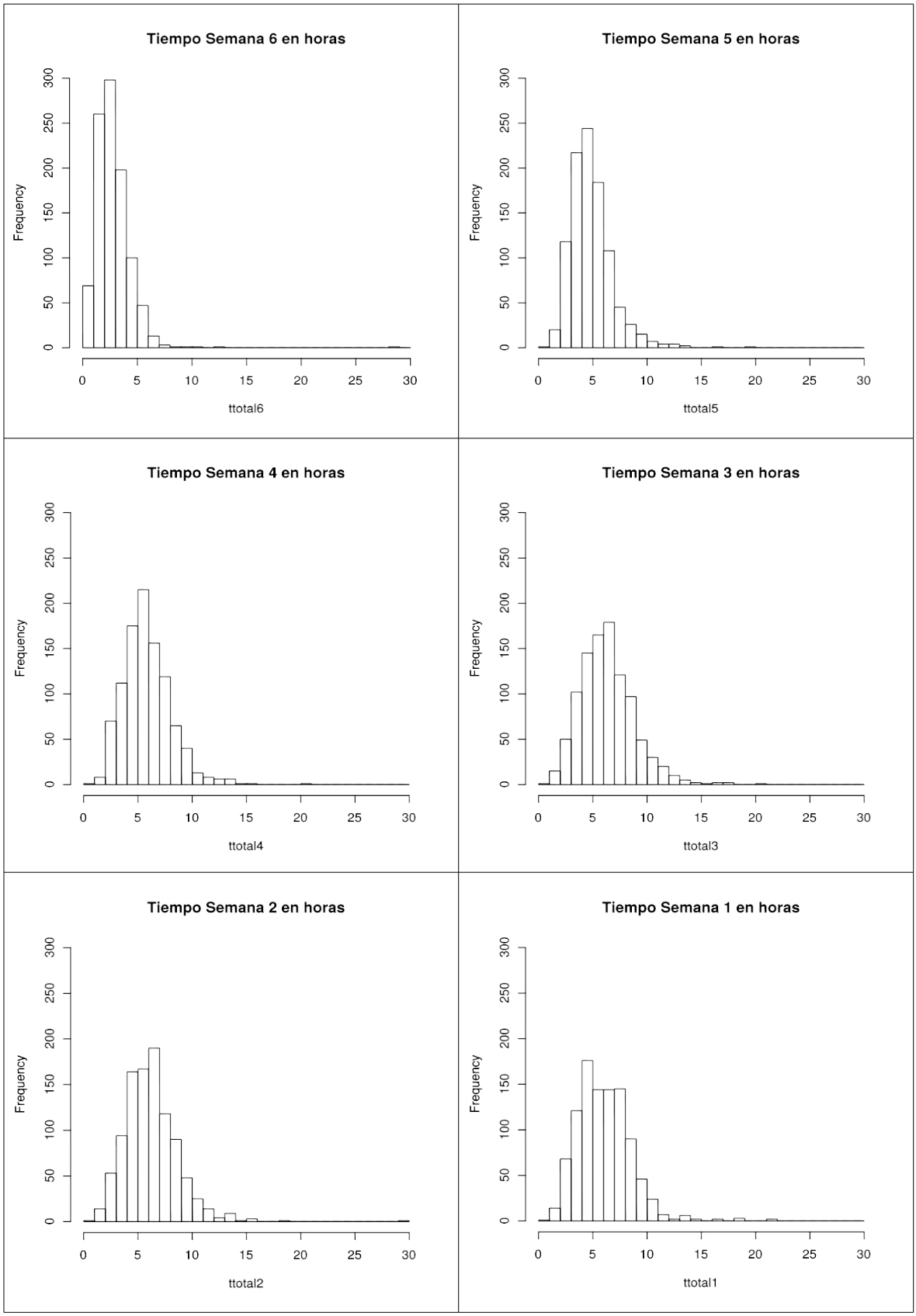

Figura 14: Tiempos totales de entrenamientos por corredor por semana Post-Filtro Londres-Boston